Universität des Saarlandes

Naturwissenschaftlich-Technische Fakultät I
Fachrichtung Informatik
Master-Studiengang Informatik

Masterarbeit

TaKG
Ein Toolkit zur automatischen Klassifikation von Gesten

vorgelegt von Robert Neßelrath

Saarbrücken, 31. März 2008

angefertigt unter der Leitung von
Prof. Dr. Dr. h.c. mult. Wolfgang Wahlster
betreut von
Dr. Jan Alexandersson

begutachtet von
Prof. Dr. Dr. h.c. mult. Wolfgang Wahlster
Dr. Jan Alexandersson

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und alle verwendeten Quellen angegeben habe.

Saarbrücken, den 31. März 2008

Einverständniserklärung

Hiermit erkläre ich mich damit einverstanden, dass meine Arbeit in den Bestand der Bibliothek der Fachrichtung Informatik aufgenommen wird.

Saarbrücken, den 31. März 2008

Danksagung

Viele Leute haben mir auf verschiedene Art und Weise geholfen, diese Arbeit fertig zu stellen. Zunächst möchte ich Prof. Wolfgang Wahlster danken, der mir die Möglichkeit gegeben hat, die Arbeit am DFKI zu schreiben.
Ein ganz besonderer Dank geht an meinen Betreuer Dr. Jan Alexandersson, der die Betreuung meiner Arbeit übernommen hat.
Ein großer Verdienst geht an die Mitarbeiter des Fachbereichs für Intelligente Benutzerschnittstellen, hierbei besonders an Norbert Pfleger und Alexander Pfalzgraf für das Korrekturlesen der Arbeit und Mira Spassova für die Betreuung meines Masterseminars.
Danke an das Altenwohnstift Egon-Reinert-Haus und an alle Probanden meiner Tests für die Evaluation.
Schlussendlich ein großer Dank an meine Mitbewohner (Julia, Markus, Marc) für die Motivation und Ablenkung *'uno mas!?'* während der Masterzeit, an Janosch für die Auflockerung der gemeinsamen Schreibleidenszeiten in der Bib und an meine Freunde und Familie für ihre bedingungslose Unterstützung.

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung eines Verfahrens zum Erlernen und Wiedererkennen von Handgesten, die mit Hilfe von Beschleunigungssensoren aufgezeichnet werden. Die Gesten sollen benutzerabhängig erlernt werden, so dass für jeden Benutzer eine eigene Wissensbasis vorliegt.

Die Entwicklung erfolgt im Rahmen des i2home-Projekts, einem Projekt zur Verbesserung der Bedienung von Geräten aus der Heimelektronik. Für die Aufzeichnung der Beschleunigungswerte wird die WiiMote von Nintendo verwendet, einem Eingabegerät für die Spielekonsole Wii, die unter anderem drei Beschleunigungssensoren enthält. Die Arbeit besteht aus vier Hauptbeiträgen.

- Zunächst wird erläutert, inwiefern Gesten in der Mensch-Maschine-Kommunikation nützlich sind und bereits verwendet werden. Es wird gezeigt, welche Formen von Gesten existieren und definiert, welche Gesten innerhalb dieser Arbeit klassifiziert werden können.
- Der zweite Beitrag beschäftigt sich mit der Klassifikation der Gesten. Es werden unterschiedliche Erkennungsverfahren und die Bibliothek Weka vorgestellt, die diese Verfahren implementiert. Weiterhin wird erörtert, weshalb eine Merkmalsextraktion nötig ist und wie diese durchgeführt wird.
- Der dritte Beitrag behandelt die Entwicklung des Toolkits TaKG. Das Toolkit bietet Entwicklern die Möglichkeit mit nur wenig Aufwand ihre Anwendung mit Gesten steuerbar zu machen.
- In der anschließenden Evaluation werden anhand von Testdaten die verschiedenen Erkennungsansätze auf ihre Präzision untersucht. Ebenfalls wird beobachtet, wie gut vor allem ältere Probanden mit der Gestensteuerung umgehen können.

Die abschließende Diskussion fasst die wesentlichen Ergebnisse der Arbeit zusammen und schlägt weitere Forschungsthemen vor.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

1.1 Multimodale Mensch-Machine Kommunikation für ältere und beeinträchtigte Menschen

Die Bedienung von neuartigen Technologien wie Mobiltelefonen, Fernsehern, Videorekordern und Computern wird meist so konzipiert, dass sie hauptsächlich von Personen verwendet werden kann, die bereits erfahren sind im Umgang mit modernen Technologien. Menschen mit kognitiven Behinderungen und auch ältere Menschen, die nicht mit neuartigen Geräten aufgewachsen sind, sind von deren Benutzung größtenteils ausgeschlossen. Der Forschungsbereich Intelligente Benutzerschnittstellen des DFKI Saarbrücken hat sich mit dem i2home Projekt[1] zum Ziel gesetzt, Küchengeräte, Geräte aus der Unterhaltungsindustrie sowie Softwareapplikationen durch innovative Bedienkonzepte auch für solche Personengruppen zugänglich zu machen und ihnen so eine Teilnahme am modernen Leben zu ermöglichen. Besonders die Bedienung von Heimelektronik soll durch intuitive Oberflächen einfacher gestaltet werden, so dass deren Verwendung ohne das vorherige seitenweise Studieren einer Bedienungsanleitung möglich wird [Alexandersson et al., 2006].

In i2home soll der Benutzer mit Hilfe eines Endgerätes, wie z.B. PDA oder Smartphone über multiple Modalitäten mit dem Zielgerät kommunizieren können. Eingabewerkzeuge sind zum Beispiel der Touchscreen oder ein Mikrofon zur sprachgesteuerten Befehlseingabe. Rückmeldungen werden visuell oder über Lautsprecher in Form von natürlicher Sprache und Tönen gegeben [Besser et al., 2007]. Ein weiterer Forschungsschwerpunkt ist das Testen neu-

[1] `http://www.i2home.org`

artiger Eingabegeräte, wie zum Beispiel ein PDA-Aufsatz für Blinde, der die Verwendung einer Brailletastatur als Eingabegerät ermöglicht.

Diese Masterarbeit beschäftigt sich damit, die multimodale Bedienung von Maschinen durch ein Konzept zu erweitern, das momentan vor allem im Spielekonsolenmarkt sehr erfolgreich ist, der Steuerung mit Gesten. Ähnliche Ansätze beinhaltet das SmartKom System das Gesten mit Infrarotkameras erfasst [Wahlster, 2003]. Besonders die Kombination von Gestensteuerung mit Sprache ermöglicht es, einen Bezug von einem sprachlich gegebenen Befehl zu einem per Geste gezeigten Objekt zu erzeugen.

In dieser Arbeit werden Bewegungen der Hand mit Hilfe von Beschleunigungssensoren gemessen und zur Bedienung von Anwendungen verwendet. Nintendo liefert mit der WiiMote für die Spielekonsole Wii einen Controller, der drei Sensoren enthält, die Bewegungen dreidimensional erfassen können. Spiele lassen sich so durch Führen des Contollers steuern, zum Beispiel teilt ein in einem Spiel vorkommender Ritter Schwerthiebe aus, die zuvor vom Spieler durch Schläge in die Luft beschrieben wurden. Für das i2home-Projekt ist von Vorteil, dass die WiiMote mit einem Preis von 40 Euro im Vergleich zu selbst entwickelten Eingabegeräten mit Beschleunigungssensoren günstig und in jedem Elektronikladen erhältlich ist.

1.2 Hauptfragen

Folgende Hauptfragen werden innerhalb dieser Masterarbeit bearbeitet.

- **Verwendung des Wii-Controllers als Eingabegerät**
  In der Masterarbeit wird untersucht, ob der Controller der Wii als Erweiterung der multimodalen Bedienung im i2home-Projekt genutzt werden kann. Hierzu muss zunächst eine Möglichkeit gefunden werden, die Eingabedaten des Controllers auszulesen und ausgeführte Gesten aufzuzeichnen.

- **Merkmalsextraktion**
  Um die Signale der gemessenen Gesten miteinander zu vergleichen, müssen Merkmale gefunden werden, durch die die Signale möglichst gut beschrieben werden können.

- **Erlernen und Wiedererkennen von Gesten**
  Da verschiedene Benutzer Gesten unterschiedlich ausführen, werden im ersten Ansatz die Gesten benutzerabhängig erlernt und klassifiziert, für verschieden Nutzer also unterschiedliche Trainingsmengen erzeugt. Hierzu

soll die Maschinenlern-Bibliothek Weka verwendet werden, deren Funktionsweise zunächst studiert werden muss. Hinzu kommt die Ermittlung eines geeigneten Klassifikators.

- **Erstellen eines Toolkits zur Klassifikation**
  Die erarbeiteten Funktionalitäten werden in einem Toolkit zusammengefasst. Es soll Schnittstellen zur Verfügung stellen, die eine schnelle Implementierung von Gestenaufzeichungs- und Gestenerkennungsvorgängen ermöglichen, um zum Beispiel Gesten zur Steuerung einer Benutzeroberfläche zu verwenden. Wichtig hierbei ist, dass das Toolkit durchgängig modularisiert ist, so dass einzelne Module einfach austauschbar sind. Hierdurch kann man das Toolkit schnell durch andere Sensoren oder Lernalgorithmen ergänzen.

- **Evaluation**
  Die Evaluation der Gestenerkennung erfolgt im Rahmen eines Wii-Workshop, bei dem Testpersonen verschiedener Alterstufen die Möglichkeit gegeben wird, eine Anwendung mit der WiiMote zu steuern. Von Interesse hierbei sind die Anwendbarkeit und Korrektheit der Gestensteuerung, insbesondere im Hinblick auf das Alter und die damit einhergehenden körperlichen Beeinträchtigungen der Testpersonen.

1.3 Aufbau der Arbeit

Kapitel 2 führt den Begriff der Geste ein, klassifiziert Gesten in unterschiedliche Typen und erläutert ihre Bedeutung für die Kommunikation. Es wird gezeigt, wie Gesten in der Mensch-Maschine-Kommunikation verwendet werden und einige Projekte vorgestellt, die sich mit der Erkennung von Gesten befassen. Zum Abschluss wird definiert, welche Art von Handgesten in dieser Arbeit klassifiziert werden können.
Kapitel 3 beschäftigt sich mit dem Maschinellen Lernen, beschreibt dessen Hauptaufgaben und stellt einige Klassifikatoren sowie die Maschinenlern-Bibliothek Weka vor.
Kapitel 4 stellt zunächst die WiiMote als Eingabegerät zur Erfassung von Gesten vor. Weiterhin beschreibt es das Aussehen der gemessenen Beschleunigungssignale und deren typischen Merkmale.
Kapitel 5 befasst sich mit der Vorverarbeitung der Beschleunigungssignale, bei der Eingabedaten für den Klassifikator erstellt werden. Es werden das Äquidistante Gitter und das Attributbeschreibende Modell zur Merkmalsextraktion vorgestellt.
In Kapitel 6 wird das für die Masterarbeit erstellte Toolkit TaKG vorgestellt. Hierbei werden die Aufgaben der verschiedenen Programmmodule erläutert und

die Kommunikation zwischen diesen beschrieben.
Die Evaluation des Toolkits erfolgt in Kapitel 7, in dem die im Rahmen der Arbeit durchgeführten Versuche und ihre Ergebnisse beschrieben werden.
Kapitel 8 fasst die Arbeit zusammen und gibt einen Ausblick auf zukünftige Forschungsrichtungen.

Kapitel 2

Kommunikation mit Gesten

Menschen kommunizieren in einer Kombination aus Sprache und Gestik. Gesten sind Teil der nonverbalen Kommunikation und werden bewusst oder unbewusst eingesetzt. Der Sozialpsychologe Mead [Mead, 1964] definierte 1932 Gesten als Zeichen in Form und Verhalten. Sie bringen einen bestimmten Sinn zum Ausdruck. Gesten wie Kopfschütteln, Vogelzeigen oder das Erheben des Zeigefingers haben also eine bestimmte Bedeutung und sollen kein beliebiges Verhalten auslösen, sondern eine ganz bestimmte Reaktion. Beim Tier erfolgt die Reaktion instinktiv oder ist aufgrund früherer Erfahrungen antrainiert. Der Mensch unterscheidet sich insofern, dass er selbst über die Art der Reaktion entscheiden kann. Elena Chalman und Sebastian Thiel geben einen Einblick in die Entwicklung von Gesten, ihre kulturgeschichtliche Bedeutung und ihren Stellenwert in Verbindung zur Sprache [Chalmann and Thiel, 2002]. Die Gestik gilt als wesentliche Entwicklungsstufe der Evolutionsgeschichte des Menschen und hat eine große Bedeutung für die Evolution der Sprache. Gestik war bereits ein Grundmechanismus der Kommunikation, bevor sich die Sprache entwickelt hat. Daher liegt nahe, dass für die intuitive Mensch-Maschine-Kommunikation, die Verwendung von Gesten eine große Bereicherung sein kann.

2.1 Gesten

In der Encyclopedia Americana ist eine Geste als *„jede bewusste oder unbewusste Körperbewegung, außer den Vokalisierungsbewegungen, durch die wir entweder mit uns selbst oder mit anderen kommunizieren“* [Encyclopedia, 2003] definiert. Die Gestik bildet somit das Ausdruckspotential des Körpers, sie kann mit Händen, Beinen, Kopf, Gesicht oder anderen Körperteilen ausgeführt werden. Gesten können einerseits redebegleitend sein und die Kommunikation un-

termauern, andererseits können sie autonom ihre eigene Aussage treffen, ohne dass es zusätzlicher sprachlicher Informationen bedarf. Michael Kipp klassifiziert Gesten wie folgt [Kipp, 2004]:

Nicht kommunikative Gesten sind instrumentelle Gesten, die verwendet werden um Objekte oder Personen zu manipulieren.

Kommunikative Gesten dienen dazu, mit einem Gegenüber zu kommunizieren. Hier wird eine weitere Unterteilung vorgenommen.

- *Symbolisch*: Diese Gesten haben eine autonome, von der Sprache unabhängige Bedeutung, häufig beschränkt auf eine Kultur. Hierzu gehören zum Beispiel das durch einen erhobenen Daumen signalisierte OK-Zeichen oder die Zeichensprache.
- *Deiktisch*: Solche Gesten werden dazu verwendet, um auf etwas zu zeigen. Das können Richtungsangaben, Objekte oder Personen sein.
- *Ikonisch*: Es werden Aussagen über ein Objekt vermittelt, etwa Größe, Form oder Orientierung.
- *Pantomimisch*: Es wird die Verwendung eines Gegenstandes oder ein Vorgang nachgeahmt, zum Beispiel das Steuern eines Lenkrads oder das Erklimmen einer Leiter.
- *Beats*: Beats sind rhythmische, sprachbegleitende Gesten, die keinen direkt Bezug zur Aussage der Rede haben sondern dazu dienen, die getroffenen Aussage zu betonen.

Gesten müssen nicht zwingend aus einer Bewegung bestehen. Durch Körper- und Handposen bewegungsfrei gezeigte Gesten werden statische Gesten genannt. Häufig wird zwischen statischen und dynamischen Gesten unterschieden.

Statische Gesten

Bei statischen Gesten ist die Haltung des Körpers oder eines Körperteils von Interesse. Ein Polizist, der durch Zeichen den Verkehr regelt nutzt zum Teil bewegungslose Gesten, etwas das seitliche Ausstrecken beider Arme als Haltebefehl. Häufig wird nur die Hand betrachtet, die durch Kombinationen von Orientierung und Beugungswinkel der Finger eine große Menge an unterschiedlichen Bedeutungen ausdrücken kann, zum Beispiel die verschiedenen Buchstaben des Alphabets (Abbildung 2.1). Statische Gesten haben meistens eine symbolische oder ikonische Bedeutung.

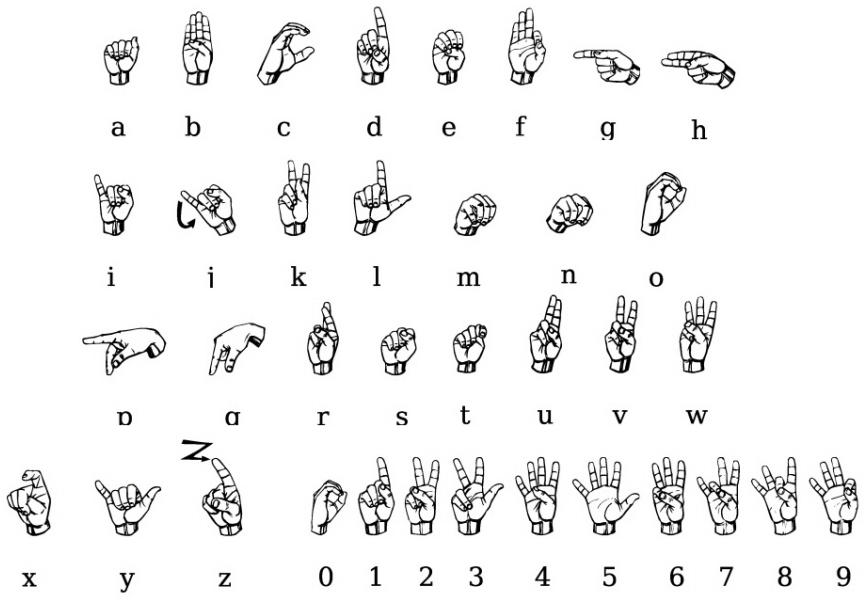


Abbildung 2.1: Fingeralphabet

Dynamische Gesten

Dynamische Gesten enthalten Bewegungen als Information. Zusätzlich kann die Haltung bestimmter Körperteile, wie das Ausstrecken eines Fingers, von Bedeutung sein. Betrachtet am Beispiel der Hand wird eine Geste durch die Orientierung der Hand, die Anordnung der Finger und den Bewegungspfad beschrieben. Die Geschwindigkeit kann eine zusätzliche Aussage treffen, z.B. vermittelt die sehr rasche Ausführung einer Bewegung eine Dringlichkeit.
Durch dynamische Gesten kann man Bewegungen und Aktionen des Lebens direkt pantomimisch nachahmen, wie das Werfen eines Balles, das Aufnehmen eines Gegenstandes oder das in die Luft Zeichnen von Zahlen und Buchstaben. Sie können auch eine symbolische Aussage haben, die nicht direkt mit einer Handlung verknüpft sein muss, etwa ein Zuwinken als Zeichen der Begrüßung.

2.2 Gestenbasierte Mensch-Maschine Kommunikation

Wie in der Einleitung des Kapitels bereis erwähnt, kann durch die Verwendung von Gesten die Kommunikation mit der Maschine für den Menschen intuitiver gestaltet werden. Viele Arbeiten beschäftigen sich bereits mit dem Thema. Billinghurst [Billinghurst, 2002] und Torres [Torres, 2007] präsentieren die Idee,

die Mensch-Maschine-Interaktion mit Hilfe von Gesten zu verbessern. Hierbei gibt es generell zwei unterschiedliche Ansätze. Einerseits können durch die Generierung von Gesten mit Hilfe virtueller Charaktere dem Benutzer Informationen lebensähnlich übermittelt werden. Anderseits kann der Nutzer selber durch alltägliche Bewegungen Maschinen steuern.

2.2.1 Gestengenerierung

Virtuelle Charaktere werden häufig in der Kommunikation von Maschine zu Mensch gebraucht. Im Virtual Human Projekt[1] werden sie als Dialogpartner entwickelt, um sie zum Beispiel als Tutoren in Bildungssoftware oder als Moderatoren zu verwenden. Je menschlicher die virtuellen Charaktere hierbei werden, desto wichtiger wird es, ihre Bewegungen möglichst realitätsgetreu und glaubwürdig darzustellen. Dies wird vor allem durch eine natürliche Gestik erreicht. Michael Kipp [Kipp, 2004] befasst sich mit der Gestengenerierung durch Imitation des Menschen. In seiner Arbeit verwendet er empirisch aus Fernsehsendungen ermittelte Daten, um natürliche sprachbegleitende Gesten zu erzeugen.

2.2.2 Gestenerkennung

Bei der Gestenerkennung soll der Computer automatisch die vom Menschen ausgeführten Gesten erkennen. Hierzu müssen die Bewegung zunächst in eine maschinenlesbare Form gebracht werden. Dies geschieht mit Hilfe von Sensoren, die die Sinnesorgane des Computers darstellen und Gesten mit Hilfe von Signalen beschreiben. Matthew Turk [Turk, 2002] gibt einen Überblick über verschiedene Techniken, Gesten aufzuzeichnen. Grundsätzlich gibt es zwei unterschiedliche Arten, die nichtinstrumentelle und die instrumentelle Gestenerkennung.

Nichtinstrumentelle Erkennung

Bei der nichtinstrumentellen Gestenerkennung werden Gesten erkannt, ohne dass der Nutzer ein Gerät halten oder bedienen muss. Dies geschieht meist durch eine oder mehrere Kameras, die Bilder von einer Person aufnehmen. Das können Aufnahmen des kompletten Körpers sein, in vielen Fällen wird nur die Hand gefilmt. Die Bilder werden mit Verfahren aus der Bildverarbeitung analysiert und so Konturen von Hand oder Körper in den Aufnahmen erkannt. Dies geschieht zum Beispiel durch das Betrachten von Farbunterschieden oder das Suchen von Kanten. Vorteil der nichtinstrumentellen Erkennung ist,

[1] `http://www.virtual-human.org/`

dass der Nutzer kein Eingabegerät benötigt und das Verfahren zum Beispiel bei Kiosksystemen gut zur Verwendung kommen kann. Nachteil ist, dass die aufgenommenen Personen sich im Bereich des Kamerablickfeldes aufhalten müssen. Weiterhin können leichte Veränderungen der Lichtverhältnisse oder des Hintergrundes zu Problemen bei der Bildanalyse führen. Dieser Effekt kann mit der Verwendung von teureren Infrarotkameras verringert werden [Wagner et al., 2006].

Instrumentelle Erkennung

Eine weitere Möglichkeit zur Gestenerkennung ist die Verwendung von Eingabegeräten. Im einfachsten Fall kann dies eine Maus sein, mit der die Gesten auf den Bildschirm gezeichnet werden. Für den Internetbrowser Firefox existiert das Plugin MouseGestures[2], das eine Steuerung des Browsers mit Hilfe von Mausgesten ermöglicht. Die Beschreibung der Geste erfolgt hierbei über die Koordinaten des Mauszeigers. Ähnliche zweidimensionale Ansätze sind mit anderen Geräten wie Touchscreens oder Joysticks möglich. Im dreidimensionalen Raum ist die Positionsermittlung nicht so einfach möglich. Hier greift man auf verschiedenste Sensoren zurück, die in Steuergeräte oder Handschuhe eingebaut werden. Beschleunigungssensoren beschreiben Gesten mit den während einer Bewegung wirkenden Beschleunigungskräften. Gyroskopsensoren messen Rotationen und Magnetometer die Ausrichtung des Eingaberätes in Bezug zum Erdmagnetfeld. Einige Beispiele für solche Geräte werden im Abschnitt 2.3 vorgestellt. Ein Vorteil der instrumentierten Geräte ist, dass sie unabhängig von Lichtverhältnissen und Position des Anwenders sind. Weiterhin müssen die Sensorinformationen vor der Verwendung nicht aufwendig analysiert werden, wie es bei Kameraaufnahmen der Fall ist. Nachteil ist der teilweise sehr hohe Preis. Sehr günstig ist die in Kapitel 4.1.1 vorgestellte WiiMote, die in dieser Arbeit als Eingabegerät verwendet wird.

2.3 Verwandte Ansätze zur gestenbasierten Mensch-Maschine-Interaktion

Menschen kommunizieren mit dem Computer größtenteils per Tastatur oder durch die Verwendung einer Maus. Hierbei werden auf dem Bildschirm dargestellte Objekte manipuliert, zum Beispiel durch das Drücken eines Knopfs oder das Ausfüllen eines Texteingabefelds. Bewegungen der Hand bewirken also eine Reaktion des Computers, doch kann man hier bereits von dem Begriff Geste sprechen? Kultenbach und Hulteen [Kurtenbach and Hulteen, 1990] verneinen

[2]Mouse Gesture Plugin, Optimoz Team, `http://optimoz.mozdev.org/gestures/index.html`

dies: *„A gesture is a motion of the body that contains information. Waving goodbye is a gesture. Pressing a key on a keyboard is not a gesture because the motion of a finger on its way to hitting a key is neither observed nor significant. All that matters is which key was pressed.“*. Nicht die Bewegung des Körpers bewirkt die Reaktion des Computers, sondern das Drücken einer Taste, ungeachtet der Tatsache, ob hierfür Finger, Nase oder Zeh verwendet wurden. Für eine durch Gesten gesteuerte Applikation sind Bewegung und Haltung des Körpers entscheidend. Befehle werden über spezielle Posen gegeben oder Bewegungspfade bzw. durch die Kombination von beidem. Das könnte zum Beispiel so aussehen, dass durch Links- und Rechtsbewegungen der Hand durch die Seiten eines Dokumentes navigiert wird. Hierbei ist ausschließlich die Bewegungsrichtung der Hand entscheidend für die Reaktion des Computers, es wird kein Objekt manipuliert.

Verschiedene Arbeiten, sowohl wissenschaftliche als auch kommerzielle, beschäftigen sich mit der Gestensteuerung von Maschinen. Im Folgenden werden einige vorgestellt.

Der Beschleunigungssensor-Handschuh

An der University of California in Berkley wird ein Sensorhandschuh zur Eingabe von Texten verwendet [Perng et al., 1999]. Hierfür wurde ein Handschuh mit sechs 2-Achsen-Beschleunigungssensoren ausgestattet (Abbildung 2.2 a). Einer der Sensoren ist auf dem Handrücken angebracht, die anderen auf den fünf Fingerspitzen. Die Sensoren messen die Beschleunigung in zwei Freiheitsgraden und ermöglichen es festzustellen, ob die jeweiligen Finger eingeknickt oder ausgestreckt sind. Bei dem Experiment geht es um die Erkennung von statischen Gesten. Es werden 28 Handzeichen trainiert, die sich in der Haltung der Finger und der Hand voneinander unterscheiden. Die Gesten beschreiben die 26 Buchstaben des Alphabets, die Leertaste und die Entfernen-Taste und dienen als Eingabebefehle für einen Texteditor. Der Nutzer kann so mit Hilfe des Handschuhs Texte schreiben und editieren. Bei Tests gelang es, 162 aufeinander folgende Gesten mit einer durchschnittlichen Geschwindigkeit von 0.91 Zeichen/Sekunde korrekt zu erkennen.

Handgestenerkennung bei einem Mobiltelefon

Eine praktische Anwendung findet die Gestenerkennung bei der Ermittlung des aktuellen Anwendungsszenarios eines Mobiltelefons [Mäntylä et al., 2000]. Ein mit Beschleunigungssensoren ausgestatteter Würfel wird an dem Mobiltelefon angebracht und nimmt statische und dynamische Gesten auf. Die statischen

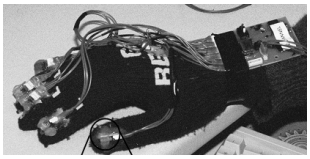
Gesten beschreiben hierbei bewegungslose Anwendungssituationen wie ein Handy, das sich auf dem Tisch oder in der Tasche befindet oder während eines Gesprächs ans Ohr gehalten wird. Als dynamische Gesten hat man sich sechs Szenarien ausgedacht, die bei der natürlichen Verwendung des Handys auftreten. Eines davon ist zum Beispiel das Aufnehmen eines klingelnden Telefons vom Schreibtisch, um sich die Nummer des Anrufers anzusehen.

Framework zur Erstellung gestengesteuerter Anwendungen

Am MIT wurde von Ari Y. Benbasant und Joseph A. Paradiso ein Framework zur Gestenerkennung und Gestennutzung in Applikation entwickelt [Benbasat and Paradiso, 2001]. Das Messinstrument hat die Form eines Würfels (Abbildung 2.2 b) und enthält zwei 2-achsige Beschleunigungssensoren und 3 Gyroskope. Die gemessenen analogen Signale werden von einem Controller erfasst, digitalisiert und kabellos an den Rechner übertragen. Das Framework ist in drei Teile aufgeteilt. Im ersten werden die Bewegungen der Sensoren aufgezeichnet, die im zweiten Teil von einem Gestenerkennungsalgorithmus analysiert werden. Hierbei werden die Bewegungen Achse für Achse auf atomare Grundbewegungen untersucht und in diese aufgeteilt. Dies können Drehungen des Eingabegerätes sein oder gerade Bewegungen. Der dritte Schritt erlaubt es einem Anwendungsentwickler, die erkannten Grundbewegungen konkurrierend und konsekutiv zu kombinieren, so dass bestimmte Kombinationen von Grundbewegungsmustern mit Ausgaberoutinen der Anwendungen verbunden werden können.

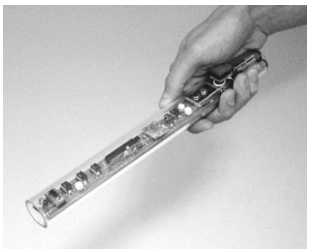
Gestenerkennung mit der XWand

Daniel Wilson von der Carnegie Mellon University und Andy Wilson von Microsoft Research haben mit der XWand (Abbildung 2.2 c) eine Fernbedienung entwickelt, die mit unterschiedlichsten Sensoren gespickt ist [Wilsona and Wilson, 2004]. Sie enthält einen 2-Achsen-Beschleunigungssensor und einen einachsigen Gyroskopen, um Drehungen entlang der vertikalen Achse zu ermitteln. Zusätzlich stellt ein 3-Achsenmagnetometer anhand des Erdmagnetsfeldes die momentane Ausrichtung der XWand fest. Die XWand hat die Form eines Stabes und lässt sich wie eine Fernbedienung in der Hand halten. Als Anwendung wird beabsichtig, Heimelektronik anhand von Gesten zu steuern. Hier wird als Beispiel das Konzept vorgestellt, Musik von den Boxen eines Gerätes aufzunehmen und zu den Boxen eines anderen Geräts zu tragen. Das Aufnehmen der Musik wird hierbei durch eine Bewegung hin zum Körper symbolisiert.



(a) Sensorhandschuh

(b) Sensorwürfel des MIT



(c) XWand

Abbildung 2.2: Messinstrumente in der Gestensteuerung

Bedienung eines Zeichenprogramms mit Handgesten

Das „Department of Numerical Analysis and Computing Science“ des Royal Institute of Technology[3] in Stockholm arbeitet an einem Projekt, bei dem über eine Kamera simultan die Bewegung einer Hand verfolgt wird und zusätzlich Handposen erkannt werden. In einer Beispielanwendung wird dies eingesetzt, um ein Zeichenprogramm zu steuern (Abbildung 2.3). Handzeichen beschreiben die Art der Aktion, die man ausführen möchte. Zum Beispiel bedeutet ein ausgestreckter Zeigefinger, dass der Cursor bewegt wird. Wird der Daumen hinzugenommen, zeichnet die Anwendung an der aktuellen Cursorposition und sind alle Finger ausgestreckt, kann man das gesamte gezeichnete Objekt auf dem Bildschirm verschieben. Die Position des Cursors wird hierbei über die Position der Hand gesteuert.

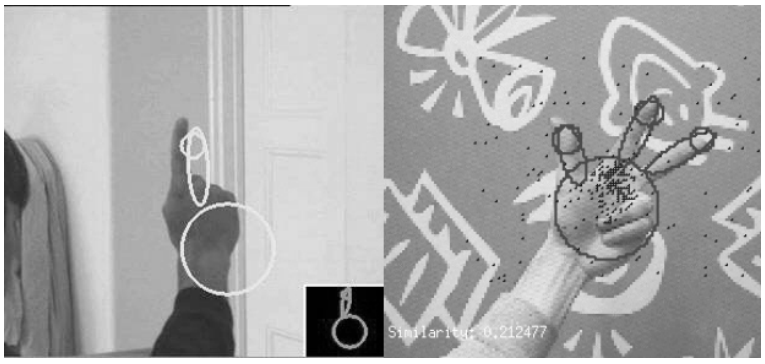


Abbildung 2.3: Handgestenanalyse per Kameraaufnahme. Hand und ausgestreckte Finger werden per Bildanalyseverfahren erkannt.

[3] `http://www.nada.kth.se/cvap/gvmdi/`

Wii-Entwicklertool LiveMove

Im kommerziellen Bereich ist die Gestenerkennung besonders durch die Wii Spielekonsole interessant geworden. Jedoch stellt die neue Technik neue Anforderungen an die Spieleprogrammierer, die sich nun zusätzlich zu den üblichen Entwicklungsaufgaben mit der Auswertung der Sensordaten beschäftigen müssen. Die Firma AiLive stellt mit dem Programm LiveMove ein Tool zur Verfügung, das es innerhalb kürzester Zeit ermöglicht, auch komplexe Gesten zu erlernen und in Spiele einzubinden [Ailive, 2006]. Spielentwickler müssen die Daten bestimmter Bewegungsabläufe nicht mehr selbst analysieren, sondern können dem Computer Gesten durch das Ausführen von Beispielbewegungen antrainieren.

2.4 Klassifizierbare Gesten in dieser Arbeit

Zum Einlesen von Gesten wird in dieser Arbeit die WiiMote mit eingebauten Beschleunigungssensoren verwendet (siehe Kapitel 4). Die Beschaffenheit des Eingabegerätes und die Art der Umsetzung des Zugriffs hierauf führen zu einigen technischen Einschränkungen, die den Typ der zur Klassifikation verwendbaren Gesten eingrenzen. Ein erster Punkt ist, dass während eines Stillstands der WiiMote keine Daten eingelesen werden. Statische Gesten fallen somit vollständig aus der Definition heraus. Weiterhin wird die WiiMote typischerweise mit der Hand bedient. Die Haltung der Finger kann nicht ermittelt werden.

Übrig bleiben dynamische Bewegungen von Arm und Hand, die sich grundsätzlich durch drei Eigenschaften voneinander unterscheiden lassen:

- Bewegungspfad
- Ausrichtung der Hand während der Bewegung
- Geschwindigkeit

Die Länge der Geste ist ebenfalls eine wichtige Eigenschaft, ist aber abhängig von Pfad und Geschwindigkeit und wird von diesen implizit mitbeschrieben.

Definition: *In dieser Arbeit klassifizierbare Gesten sind Bewegungen der Hand, die anhand des relativen Bewegungspfades zur Startposition, der Ausrichtung der in der Hand gehaltenen WiiMote während der Bewegung und der Geschwindigkeit der Bewegung beschrieben und differenziert werden können.*

Die Körperhaltung der Person kann nicht ermittelt werden, jedoch kann sie Auswirkungen auf die Messungen der WiiMote haben. Streckt eine Person ihre linke Hand nach links vom Körper weg, einmal im Stehen und einmal im Liegen, unterscheiden sich für die WiiMote beide Bewegungen anhand von Pfad der Bewegung und Ausrichtung des Gerätes, die Messwerte sind also unterschiedlich (Abbildung 2.4). Für die Person selbst ist die Bewegung die gleiche, nur dass die Körperhaltung verschieden ist.

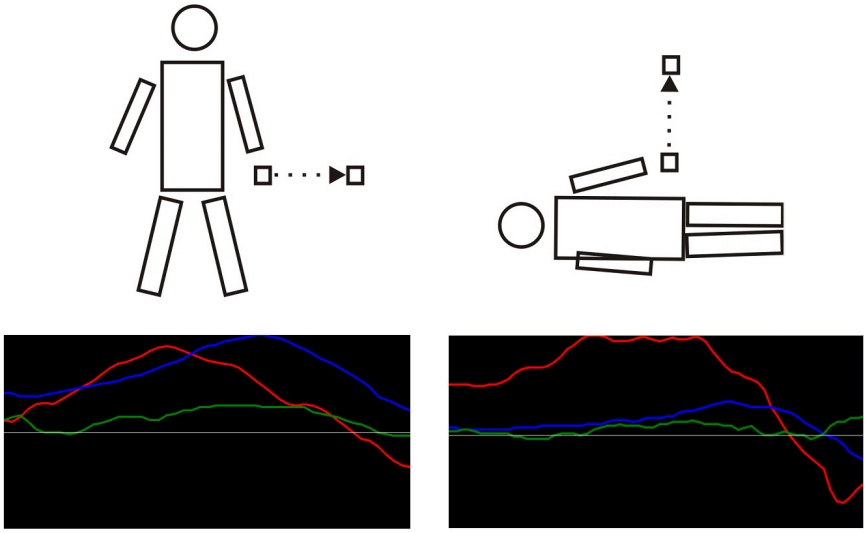


Abbildung 2.4: Die Person beschreibt mit der Geste ein Bewegung der linken Hand nach links vom Körper weg, im Stehen und im Liegen. Die Messungen der WiiMote unterscheiden sich hierbei vor allem im Verlauf der blauen Kurve.

Es bleibt zu klären, ob beide Bewegungen als ein und dieselbe Geste aufgefasst werden sollen. Abhilfe schafft der Umstand, dass die Bewegungen zu den jeweiligen Gestenklassen erlernt werden, wodurch die Lösung der obigen Fragestellung auf die Anwendungsfälle angepasst werden kann. Beim Erlernen der Gesten können so beide Bewegungen zu einer Gestenklasse zusammenfasst werden oder es werden zwei unterschiedliche Gestenklassen erzeugt. So können beide Gesten als Linksbewegung aufgefasst werden, es kann jedoch auch zwischen Linksbewegung im Stehen und im Liegen unterschieden werden. Der Lernprozess ermöglicht weiterhin, Bewegungen auf identischen Bewegungspfaden aufgrund ihrer Geschwindigkeit zu differenzieren. So können eine langsame und eine schnelle Kreisbewegung entweder als unterschiedliche Gesten aufgefasst werden oder man fasst beide unter dem Begriff der Kreisbewegung zusammen.

Mit dem hier vorgestellten Ansatz können symbolische und pantomimische Gesten erkannt werden, wenn sie dynamisch sind und mit der Hand ausgeführt werden. Das Erkennen ikonischer Gesten ist nur zum Teil möglich. Es können z.B.

mit Bewegungen Formen wie z.B. ein Kreis beschrieben werden, die genaue Beschreibung von Größen ist jedoch schwierig. Für Deiktische Gesten eignen sich die Beschleunigungssensoren wenig, da keine Informationen zur Position der WiiMote im Raum erfasst werden können. Durch Kombination der dynamischen Gestenerkennung mit der Positionsermittlung der WiiMote über die eingebauten Infrarotsensoren würde dies prinzipiell machbar werden.

2.5 Zusammenfassung

Kommunikative Gesten sind Körperbewegungen, außer den Vokalisierungsbewegungen, mit denen wir kommunizieren. Sie können in fünf Klassen unterteilt werden: symbolisch, deiktisch, ikonisch, pantomimisch und Beats. Weiterhin kann man zwischen statischen und dynamischen Gesten unterscheiden. In der Kommunikation zwischen Mensch und Maschine kommen Gesten vermehrt zum Einsatz. Einerseits werden Gesten generiert, um virtuelle Charaktere in der Kommunikation mit dem Menschen realistischer wirken zu lassen, andererseits erkennt der Computer Gesten und lässt sich durch diese steuern. Einige Arbeiten hierzu wurden in Abschnitt 2.3 vorgestellt. Als Eingabe dienen bei nicht instrumentierten Anwendungen Kameraaufnahmen, bei instrumentierten Anwendungen Sensoren, wie zum Beispiel Beschleunigungssensoren, die in Eingabegeräte oder Handschuhe eingebaut werden. In der vorliegenden Arbeit wurde die WiiMote als Eingabegerät verwendet, durch deren technische Eigenschaften die in dieser Arbeit klassifizierbaren Gesten auf Bewegungen der Hand beschränkt wurden. Es können symbolische, pantomimische und begrenzt ikonische Gesten erkannt werden.

Kapitel 3

Maschinelles Lernen

Maschinelles Lernen soll automatisch Wissen anhand von Erfahrungen generieren. In diesem Kapitel wird zunächst der Vorgang des Maschinellen Lernens mit dem Spezialfall des Klassifizierens vorgestellt. Im Anschluss werden die Ideen hinter den Funktionsweisen der Klassifikationsverfahren Neuronale Netze und Support Vector Machine erklärt und die Java Bibliothek Weka vorgestellt, die ein umfangreiches Programmpaket für das Maschinelle Lernen zur Verfügung stellt.

3.1 Einführung in das Maschinelle Lernen

Die Künstliche Intelligenz (KI) ist ein Teilgebiet der Informatik und beschäftigt sich damit, intelligentes Verhalten am Computer nachzuahmen und zu automatisieren. Einer der wichtigsten Aspekte von Intelligenz ist die Lernfähigkeit. Hiermit beschäftigt sich das Maschinelle Lernen. Häufig werden in der KI menschliche kognitive Prozesse verwendet und auf computergestützte Umgebungen angepasst, um komplexe Probleme zu lösen. Beim Maschinellen Lernen versucht man, natürliche Vorgänge des Lernprozesses, zum Beispiel im menschlichen Gehirn, auf rechnerbasierte Systeme abzubilden [Mitchell, 1997].

Maschinelles Lernen benötigt Erfahrung, bestehend aus Daten, die Beispiele oder Messungen zu einem Problem sind. Diese werden in der Lern- bzw. Trainingsphase auf Gesetzmäßigkeiten untersucht und so Regeln erstellt, die die Beispiele verallgemeinern. Die so erlernten Muster werden für den Rechner in Entscheidungsfunktionen umgesetzt und ermöglichen es, auch neue und unbekannte Daten automatisch möglichst korrekt zu beurteilen.

Für das Maschinelle Lernen gibt es ein weites Spektrum an möglichen Anwen-

dungen, z.B. medizinische Analysen, Aktienmarktanalysen, Aufdecken von Kreditkartenbetrug, Klassifikation von DNA-Sequenzen oder Sprach- und Schrifterkennung. Besonders zuletzt genannte Themen sind für diese Arbeit wichtig, da sie eng mit der Gestenerkennung verwandt sind. Sowohl bei der Sprach- als auch bei der Schrifterkennung müssen Signale bestimmten Worten, Wortteilen oder Buchstaben zugeordnet werden. Bei der Gestenerkennung müssen Signale, im Falle dieser Masterarbeit Signale von Beschleunigungssensoren, Gesten zugeordnet werden.

3.2 Automatische Klassifikation

Beim Maschinellen Lernen wird zwischen überwachtem und unüberwachtem Lernen unterschieden.

Während des unüberwachten Lernens stehen während der Lernphase keine Informationen über die erwünschten Ausgaben zu den erlernten Eingabedaten zur Verfügung. Beim *Clustering*, einer Form des unüberwachten Lernens, werden Gemeinsamkeiten zwischen den einzelnen Daten gesucht und ähnliche Objekte zu Clustern zusammengefasst. Nach der Lernphase können neue Objekte den so erzeugten Clustern zugeordnet werden. Das unüberwachte Lernen wird in dieser Arbeit nicht verwendet.

Bei überwachten Lernverfahren steht Expertenwissen zur Verfügung, dem die zugehörigen Ausgaben zu den Trainingsobjekten bekannt sind. Bei der Klassifikation wird zu dem Eingabedaten jedes Trainingsobjekts eine Klasse angegeben, der das Objekt zuzuordnen ist. Es werden wiederum Gemeinsamkeiten gesucht und mit deren Hilfe Regeln erstellt, die automatisch neue Objekte einer Klasse zuweisen können. Folgendes Beispiel stellt die Diagnose als klassisches Klassifikationsproblem vor.

Es soll ein Verfahren entwickelt werden, das Ärzte bei der Diagnose von Krankheiten unterstützt. Hierbei ist es wichtig, Patienten mit den einzelnen Krankheiten zu finden, um Zusammenhänge zwischen den bei ihnen ermittelten Werten und Symptomen und den Krankheiten festzustellen. Selbstverständlich müssen auch gesunde Patienten zu dieser Studie herangezogen zu werden, um einen Aufschluss darüber zu erhalten, welche Werte im normalen Bereich liegen. Nach Analyse der Daten soll es möglich sein, die Krankheit neuer Patienten aufgrund ihrer Werte zu bestimmen oder zumindest die Anzahl möglicher Krankheiten einzuschränken.

Die Daten zu den einzelnen Patienten dienen als Wissensbasis und werden beim Maschinellen Lernen *Trainingsmenge* genannt. Eine Trainingsmenge besteht aus

Objekten. Im Beispiel sind dies die Patienten. Beschrieben werden die Objekte durch *Attributsvektoren*. Im Falle der Patienten sind dies die Angaben zu gemessenen Werten oder beobachteten Symptome, wie z.B. Fieber, Kopfschmerzen oder erhöhter Puls. Das erworbene Wissen über die Diagnose wird über eine *Klassifizierungsfunktion* beschrieben. Mit Hilfe der Klassifizierungsfunktion können neue, noch nicht behandelte Patienten auf Krankheiten diagnostiziert werden. Die Krankheit, die diagnostiziert wird, ist hierbei die *Klasse*, die das Klassifikationsverfahren ermittelt. Im optimalen Fall wird eine Entscheidungsfunktion entwickelt, die eine fehlerfreie Diagnose von Patienten ermöglicht.

3.3 Klassifikationsverfahren

Es gibt eine große Anzahl an Mustererkennungsverfahren, darunter Entscheidungsbäume, Hidden Markov Models oder Naive-Bayes-Klassifizierer. Im Folgenden werden die beiden in der Arbeit verwendeten Klassifikatoren kurz vorgestellt.

3.3.1 Künstliche Neuronale Netze

Die Idee des Künstlichen Neuronalen Netzes entstammt zum Teil der Beobachtung von biologischen Lernsystemen. Natürliche Neuronale Systeme im Gehirn bestehen aus einem sehr komplexen Netz aus Nervenzellen (Neuronen) die auf chemischem und elektrischem Weg Informationen untereinander austauschen. Ihre Aktivitäten werden typischerweise durch Verbindungen zu anderen Neuronen angeregt bzw. gehemmt, wobei ein Neuron im menschlichen Gehirn im Schnitt mit zehntausend anderen Neuronen vernetzt ist. Neuronen haben sowohl Eingangsverbindungen, über die sie Reize anderer Neuronen aufnehmen, als auch Ausgangsverbindungen, die Reize feuern sobald ein gewisser Reizschwellwert überschritten wird. Neuronale Netze ermöglichen es, komplexe Muster zu erlernen ohne deren zugrunde liegenden Regeln abstrahieren zu müssen.

Künstliche Neuronale Netze bestehen aus Knoten, die die Neuronen durch mathematische Modelle nachbilden, die reellwertige Eingaben zu reellwertigen Ausgaben verarbeiten können. Produzierte Ausgaben können wiederum als Eingabe für andere Knoten dienen, die Verbindungen zwischen den Knoten sind hierbei gewichtet. Abbildung 3.1 stellt ein vereinfachtes Neuronales Netz dar. Während der Trainingsphase kann das Netz durch unterschiedliche Manipulationen lernen:

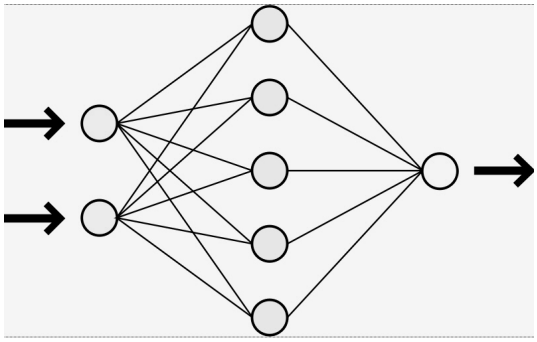


Abbildung 3.1: Vereinfachte Darstellung eines neuronalen Netzes. Die Eingabe wird in den ersten beiden Knoten verarbeitet und deren Ausgabe wiederum als Eingabe für weitere Knoten verwendet. Dieser Vorgang setzt sich fort, bis der letzte Knote der Kette ein Signal liefert. (Quelle: Wikipedia)

- Erzeugen und Löschen von Knoten
- Erzeugen und Entfernen von Verbindungen zwischen Knoten
- Anpassung des Schwellenwertes eines Knotens
- Anpassung der Gewichtung w_{ij} der Verbindung zwischen Knoten i zu Knoten j

Die verschiedenen Netztypen unterscheiden sich zum größten Teil in ihren Netztopologien. Es gibt einschichtige und mehrschichtige Netze. In dieser Arbeit wird ein mehrschichtiges Netz verwendet. Hierbei sind alle Knoten in Schichten eingeteilt, wobei ein Knoten einer Schicht immer mit allen Knoten der nächsten Schicht mit einer bestimmten Gewichtung verbunden ist. Zur vorherigen Schicht gibt es keine Verbindung und auch keine Verbindungen, die Schichten überspringen. Weiterhin sind die Verbindungen unidirektional und nicht zyklisch. Das Netz ist ein Feedforward-Modell. Es hat eine Eingabeschicht, in der Knoten enthalten sind, die nur Signale weiterleiten ohne von anderen Neuronen welche zu empfangen und eine Ausgabeschicht, deren Knoten nur Signale empfangen und nicht weiterleiten. Die zwischen Ein- und Ausgabeschicht liegenen Knoten werden meist in einer oder mehreren versteckten Schichte angeordnet [Bishop, 1995].

Neuronale Netze sind besonders für Probleme geeignet, deren Trainingsdaten sich auf komplexe Sensordaten beziehen wie zum Beispiel Aufnahmen von Kameras oder Mikrofonen. Die Objekte können hierbei durch viele Attribute beschrieben werden und jeden reellen Wert annehmen. Das Verfahren ist ziemlich robust gegenüber Fehlern, so dass die Trainingsdaten Rauschen enthalten dürfen. Der Lernprozess kann je nach Umfang der Trainingsdaten eine längere

Zeit in Anspruch nehmen, die Auswertung der erlernten Zielfunktion erfolgt jedoch normalerweise sehr schnell.

Künstliche Neuronale Netze sind besonders für Anwendungen interessant, für deren zu lösende Probleme nur wenig explizites systematisches Wissen vorliegt. Hierzu gehören Text-, Sprach-, und Bilderkennung [Mitchell, 1997].

3.3.2 Support Vector Machine

Die Support Vector Machine ist ein Klassifikator, bei dem die Menge an Objekten so in ihre Klassen unterteilt wird, dass der Bereich um die Klassengrenzen herum, in dem keine Trainingsobjekte enthalten sind, maximiert wird. Dies löst man, indem man den Eigenschaftsraum mit einer Hyperebene in ihre Klassen unterteilt. Hierbei soll der Abstand zwischen den am nächsten an der Grenze liegenden Objekten und der Grenze selbst minimiert werden. Die Hyperebene ist die Lösung der Gleichung:

$$\underset{\mathbf{w}\in\mathcal{H},b\in\mathbb{R}}{maximize}\ min\{\|\mathbf{x}-\mathbf{x}_i\||\mathbf{x}\in\mathcal{H},\langle\mathbf{w},\mathbf{x}\rangle+b=0,i=1,...m\}$$

Die Gleichung besagt, dass die freien Variablen $\mathbf{w}$ und b der Hyperebene so gewählt werden sollen, dass das Objekt $\mathbf{x}_i$, das von den Objekten $\mathbf{x}_1,..,\mathbf{x}_m$ den kleinsten Abstand zu einem Punkt $\mathbf{x}$ auf der Hyperebene $\langle\mathbf{w},\mathbf{x}\rangle+b=0$ hat, maximiert werden soll. Die Vektoren, die den geringsten Abstand haben, werden hierbei Stützvektoren genannt.

Ein Problem bei dieser Trennmethode ist, dass die Hyperebene nicht gebogen werden darf, wodurch die Objekte linear trennbar sein müssen, was in der realen Anwendung jedoch selten der Fall ist.

Deswegen wird der Vektorraum in einen höherdimensionalen Vektorraum überführt (siehe Abbildung 3.2). Dies kann soweit gehen, dass man unendlich viele Dimensionen verwendet. Im höherdimensionalen Raum ist es nun möglich, eine linear trennende Hyperebene zu finden. Durch Rücktransformation der Hyperebene in den ursprünglichen Attributraum erhält man meistens eine nicht linear trennende Hyperebene. Hierbei kann es vorkommen, dass die entstehenden Hyperflächen nicht einmal mehr zusammenhängend sind.

Der Rechenaufwand der Transformation kann groß werden, da jeder Punkt in den höherdimensionalen Raum transformiert und anschließend wieder rücktransformiert werden muss. Dies kann man mit Hilfe des Kernel-Tricks umgangen werden. Dieser definiert eine Funktion k so, dass gilt:

$$k(x,x')=\langle\mathbf{x},\mathbf{x}'\rangle$$

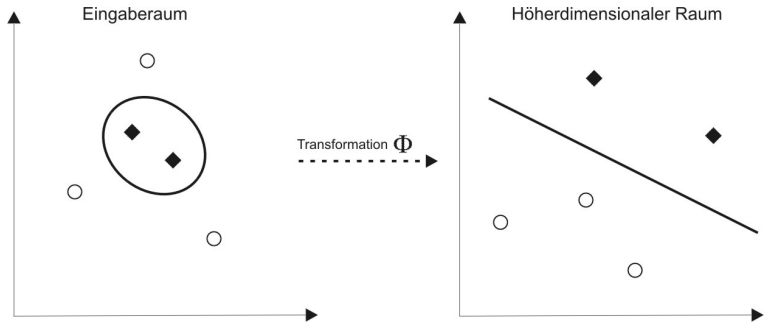


Abbildung 3.2: Die Trainingsdaten werden in einen höher dimensionalen Eigenschaftsraum transformiert und eine lineare Hyperebene gesucht, die sie mit maximalem Abstand trennt. Dies führt in den meisten Fällen zu einer nichtlinearen Grenze im Eingaberaum, in diesem Fall einem Kreis (Quelle [Schölkopf and Smola, 2002]).

Hierbei sind x, x' die Objekte im Eingaberaum und $\mathbf{x}, \mathbf{x}'$ die in den höherdimensionalen Raum transformierten Objekte. Da die transformierten Werte bei der Bestimmung der optimalen Hyperebene nur im Skalarprodukt vorkommen, kann durch den Kernel-Trick die Transformation in den höherdimensionalen Raum erspart und direkt mit den Objekten aus dem Eingaberaum gerechnet werden.

Im Vergleich zu neuronalen Techniken zeichnen sich SVM durch eine geringe Gefahr der Überoptimierung und durch die Konvergenz zu einem eindeutigen globalen Optimum aus (im Gegensatz zu Backpropagation- und ähnlichen Algorithmen, die durch einen eingebauten Zufallsgenerator immer wieder andere Ergebnisse nach einem neuen Training mit gleichen Trainingsdaten liefern).

Die SVM ist theoretisch fundiert und im Gegensatz zu herkömmlichen neuronalen Netzen leicht handhabbar. Vorhandene Lernalgorithmen sind schnell und zeigen überragende experimentelle Ergebnisse, die sich auch in der Praxis als brauchbar erwiesen haben [Schölkopf and Smola, 2002].

In Arbeit wird der *Sequential Minimal Optimization* Algorithmus von John C. Platt benutzt mit dem ein Support-Vector-Klassifizierer trainiert wird, der einen polynomialen skalierten Kern verwendet [Platt, 1998].

3.4 Weka

Weka[1] (Waikato Environment for Knowlegde Analysis) ist eine Sammlung von in Java implementierten Maschinenlern-Algorithmen und wird an der University of Waikato in Neuseeland entwickelt. Weka soll die Anwendung maschineller Lernverfahren für eine Vielzahl von realen Problemen einfacher und intuitiver gestalten. Die Bibliothek umfasst eine große Anzahl der gängigsten Algorithmen zu den grundlegenden Funktionalitäten im Maschinenlernen: Klassifikation, Regressionsvorhersage, Clustering und Attributsauswahl. Als Klassifikatoren sind zum Beispiel Entscheidungsbäume, Naive Bayes, Support Vector Machine und Neuronale Netze enthalten, als Clusterer SimpleKMeans, Farthest First und EM Clustering. Weiterhin werden Module zur Vorverarbeitung und Visualisierung von Daten, zur Validierung und Visualisierung von trainierten Modellen und Anbindung von Datenbanken zur Verfügung gestellt.

3.4.1 Gründe für die Verwendung von Weka

Das Weka-Paket ist quelloffen und daher frei erhältlich und verwendbar. Es ist in der Praxis erprobt, getestet und größtenteils ausgereift. Weiterhin bietet es alle nötigen Klassifikationsalgorithmen für diese Arbeit und ist gut dokumentiert. Die Anbindung kann direkt über Java Bibliotheken erfolgen und Weka so in den eigenen Programmcode integriert werden. Der Umweg über andere Applikationen bleibt erspart. Eine Alternative zu Weka ist die in C++ implementierte Maschinenlern-Bibliothek MLC++ [Kohavi et al., 2006].

3.4.2 Verwendung von Weka

Die Lernschemata von Weka sind komplett in Java implementiert und in Paketen organisiert. Es gibt unterschiedliche Aufrufmöglichkeit. Eine erfolgt direkt über die Kommandozeile. Hierbei ruft der Java Interpreter die gewünschten Pakete auf, zum Beispiel einen Naive Bayes Klassifikator. Informationen über Optionen und zu bearbeitende Dateien werden hier als Parameter dem Kommandozeilenaufruf angefügt.

Wesentlich komfortabler ist die Bedienung über die zu Weka gehörende grafische Benutzeroberfläche. Der Benutzer wird hierbei beim Öffnen, Vorverarbeiten, Klassifizieren und Visualisieren der Daten unterstützt, indem er die gewünschten Aktionen über Menüs auswählen kann. Weiterhin gibt der so ge-

[1] `http://www.cs.waikato.ac.nz/~ml/weka/`

nannte Experimenter ihm die Möglichkeit, die Ergebnisse verschiedener Klassifikatoren und Einstellungen zu vergleichen [Holmes et al., 1994].

Der volle Funktionsumfang von Weka kann auch in einer eigenen Java Anwendung verwendet werden. Mit dem Einbinden des Java Pakets `weka.jar` hat man Zugriff auf sämtliche Weka-Module und kann so anhand weniger Methodenaufrufe aus Trainingsdaten Klassifikatoren erzeugen und mit ihnen neue Instanzen klassifizieren. Somit erhält man die Möglichkeit, mit nur geringem Aufwand Maschinelles Lernen in die eigene Anwendung zu integrieren.

3.4.3 Eingabeobjekte in Weka

Die Eingabe bei Weka besteht aus Konzepten, Instanzen und Attributen. Das Konzept beschreibt das maschinell zu erlernende Problem. Es definiert Attribute, durch die ein Objekt beschrieben wird, und die Ausgabeattribute, denen die Objekte zugeordnet werden können. Zulässig sind sowohl nominale als auch numerische Attribute. Die Ausgabeattribute sind in den meisten Fällen nominal, nämlich die Namen der Klassen, denen die Instanzen zugeordnet werden. Die Trainingsmenge der Lernphase besteht aus Instanzen, die Paare aus Attributvektoren und Ausgabewerten bilden.

Beispiel 3.1(Weka - Source)

In folgendem Beispiel [Witten and Frank, 2000, S. 10-12] besteht das Konzept aus der Entscheidung, bei der aktuellen Wettersituation draußen zu spielen oder nicht.

- Konzeptbezeichnung

```
@relation weather
```

- definierte Attribute und mögliche Attributwerte

```
@attribute outlook sunny, overcast, rainy
@attribute temperature real
@attribute humidity real
@attribute windy TRUE, FALSE
```

- Klassenattribut (Ausgabeattribut)

```
@attribute play yes, no
```

- Trainingsinstanzen beschrieben durch ihre Merkmale

```
@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
...
```

Tabelle 3.1: Tabellarische Darstellung des Wetterbeispiels

Aussicht	Temperatur	Luftfeuchtigkeit	Wind	Spielen
sonnig	85.0	85.0	FALSCH	nein
sonnig	80.0	90.0	WAHR	nein
bewölkt	83.0	86.0	FALSCH	ja
regnerisch	70.0	96.0	FALSCH	ja
regnerisch	68.0	80.0	FALSCH	ja
regnerisch	65.0	70.0	WAHR	nein
bewölkt	64.0	65.0	WAHR	ja
sonnig	72.0	95.0	FALSCH	nein
sonnig	69.0	70.0	FALSCH	ja
regnerisch	75.0	80.0	FALSCH	ja
sonnig	75.0	70.0	WAHR	ja
bewölkt	72.0	90.0	WAHR	ja
bewölkt	81.0	75.0	FALSCH	ja
regnerisch	71.0	91.0	WAHR	nein

Bei Betrachtung des Beispiels kann man ausgehend von den Instanzen annehmen, dass bei bewölktem Wetter immer gespielt wird. Scheint die Sonne, ist keine direkte Aussage möglich. Wirft man jedoch einen zusätzlichen Blick auf die anderen Attribute, findet man eine feinere Unterteilung basierend auf der Luftfeuchtigkeit. Bei einem Wert kleiner 75 kann gespielt werden, bei einem Wert größer nicht. Auf diese Weise entwickelt sich nach und nach ein Muster, das die Entscheidungsfindung beschreibt. Zeichnet man diese Entscheidungen in einem Baum auf, so hat man einen ersten Lernalgorithmus, den Entscheidungsbaum (siehe Abbildung 3.3).

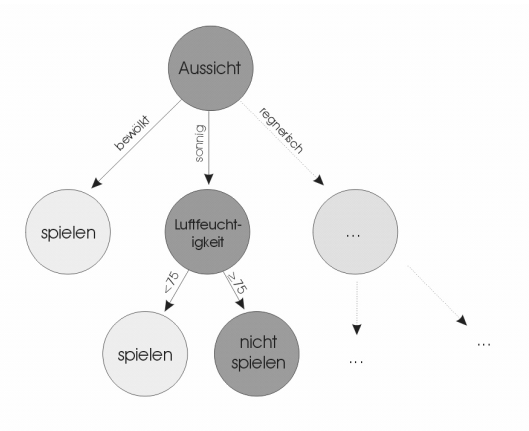


Abbildung 3.3: Ausschnitt aus dem Entscheidungsbaum für das Wetterbeispiel. Bei bewölktem Wetter wird direkt die Entscheidung getroffen, dass gespielt werden kann. Ist es sonnig, wird zusätzlich überprüft, dass die Luftfeuchtigkeit nicht zu hoch ist. Bei regnerischem Wetter wird der Entscheidungsprozess in ähnlicher Weise fortgesetzt.

3.5 Zusammenfassung

In diesem Kapitel wurde die Grundidee des Maschinellen Lernens vorgestellt, dem automatischen Erzeugen von Entscheidungsfunktionen auf der Basis von vorgegebenem Wissen. Diese können Vermutungen zu neuen, unbekannten Daten aufstellen. Besonders wurde auf die Klassifikation eingegangen, deren Ziel es ist, Dateninstanzen automatisch speziellen Klassen zuzuordnen. Anhand eines Beispiels, in dem automatisch Patienten diagnostiziert werden, wurde dieser Vorgang erläutert. Im Anschluss wurden die zwei im Rahmen der Arbeit verwendete Maschinenlernalgorithmen vorgestellt und ihre Funktionsweise erklärt: Das Vorgänge im Gehirn nachahmende Neuronale Netz und die Support Vector Machine. Im letzten Abschnitt des Kapitels wurde das Softwarepaket Weka vorgestellt. Weka ist quelloffen und implementiert die gängigsten Maschinenlernverfahren. Diese werden in einer Java-Bibliothek zur Verfügung gestellt und sind leicht in den eigenen Code integrierbar. Weka wurde für die Implementierung des Toolkits TaKG verwendet.

Kapitel 4

Beschleunigungssensoren

Der Mensch nimmt seine Umwelt über Rezeptoren wahr. Diese spezialisierten Zellen bringen externe und interne chemische oder physische Reize in eine für das Nervensystem verständliche Form. Am Computer wird dieser Vorgang mit Hilfe von Sensoren nachgeahmt. In dieser Arbeit werden Bewegungen mit Hilfe von Beschleunigungssensoren erfasst. Dieses Kapitel stellt das hierzu verwendete Eingabegerät, die WiiMote, vor und beschreibt die von ihr erzeugten Sensorsignale.

In Kapitel 2.3 wurden einige Arbeiten vorgestellt, in denen man unter Verwendung von Beschleunigungssensoren Gesten erlernt und wiedererkennt. Die Hardware hierzu wurd meist eigens für die Projekte konzipiert. Bestandteil sind nicht ausschließlich Beschleunigungssensoren sondern auch andere, die Informationen ergänzende Sensoren. Dies können z.B. Gyroskope sein, die die Rotation des Gerätes messen oder Magnetometer, mit deren Hilfe die Ausrichtung des Eingabegeräts bezogen auf das Erdmagnetfeld ermittelt wird.

Selbst erbaute Messinstrumente haben den Nachteil, dass Entwicklung und Zusammenbau Zeit benötigen und Kosten verursachen, die bis in den dreistelligen Bereich gehen können. Eine Alternative hierzu bietet die WiiMote von Nintendo, die in großen Mengen produziert wird und daher günstig zu erwerben ist. Sie ist in beinahe jedem Elektronikhandel erhältlich und weit verbreitet, was sie zu einem interessanten Eingabewerkzeug für die Gestenerkennung macht.

4.1 Überblick über die Nintendo Wii

Im Jahr 2006 hat Nintendo seine neue Spielkonsole, die Wii, herausgebracht und mit seinen Verkaufszahlen die direkten Konkurrenten Microsoft (Xbox 360) und

Sony (Playstation 3) deutlich geschlagen. Ein wesentlicher Grund hierfür ist ihr neuartiges Bedienkonzept. Beschleunigungssensoren ermöglichen es, die Bewegungen des Controllers auf die Spielfiguren umzusetzen, so dass die Steuerung nicht mehr ausschliesslich durch das Drücken von Knöpfen auf dem Controller, sondern durch die Bewegung des Conrollers selbst erfolgt.

4.1.1 WiiMote

Vom Aussehen her ist die WiiMote einer Fernbedienung ähnlich (Abbildung 4.1) und bietet neben acht Knöpfen ein Steuerkreuz als Eingabemöglichkeit. Rückmeldungen an den Benutzer können entweder über einen eingebauten Lautsprecher, vier LED-Leuchten oder das eingebaute, Vibrationseffekte erzeugende ‚Rumble Pack' gegeben werden.

Innovativ sind zwei neue Arten der Spielsteuerung. An der Front des Controllers ist ein Infrarotsensor angebracht, die zwei Referenzpunkte einer am Bildschirm befindlichen Infrarotleiste erkennen kann und so in der Lage ist, die Position des Controllers im Raum zu ermitteln. Somit ist es möglich, mit einer mit Mauszeigern vergleichbaren Präzision, Menüpunkte oder Spielobjekte auf dem Bildschirm anzuvisieren. Die zweite Neuheit sind drei in den Controller integrierte Beschleunigungssensoren. Diese ermöglichen das Erfassen von Bewegungen und Drehungen des Controllers, wodurch Bewegungen des Spielers direkt auf ein Spiel übertragen werden können, zum Beispiel das Schwingen eines Baseballschlägers oder das Führen eines Schwertes. Weiterhin geben die Sensoren bei ruhendem Controller Informationen über seine Ausrichtung an, da die Schwerkraft permanent eine Kraft ausgeübt, die je nach Haltung der WiiMote auf die drei Sensoren verteilt wird.

Am unteren Teil des Controllers befindet sich ein Anschluss für Erweiterungsgeräte, die zusätzliche Eingabemöglichkeiten bieten. Hierfür gibt es diverse Geräte, zum Beispiel ein Controller im Stil von klassischen Controllern, der neben Steuerkreuz und Knöpfen einen Joystick integriert hat. Alternativ gibt es den Nunchunk-Controller mit Joystick, 2 Knöpfen und zusätzlichen Beschleunigungssensoren.

4.1.2 Kommunikation der WiiMote mit dem PC

Die Kommunikation mit der Konsole erfolgt kabellos über Bluetooth. Als Bluetooth-Controller wird der 'BCM2042 Advanced Wireless Keyboard/Mouse Bluetooth® Chi' verwendet, der dem Bluetooth Human Interface Device (HID) Standard folgt und auch in kabellosen Mäusen und Tastaturen eingebaut ist.

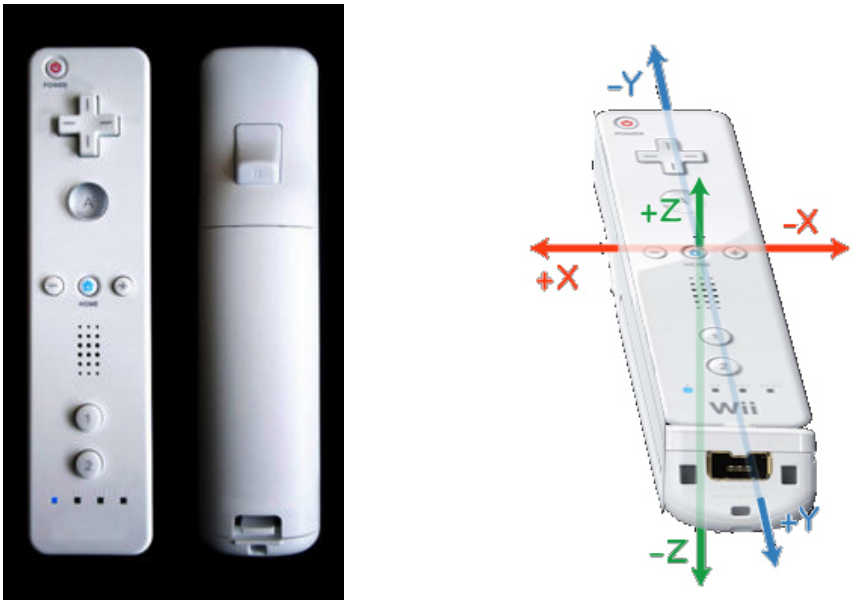


Abbildung 4.1: **Links:** Vor- und Rückseite der WiiMote. **Rechts:** Die farbigen Pfeile beschreiben die drei Achsen der Beschleunigungssensoren

Über Bluetooth versandte Informationen können auch an einem PC ausgelesen werden.

Einige Programme benutzen die WiiMote als Eingabegerät für den PC. Erste Ansätze ermöglichten es, die Eingabeinformationen zum Steuern des Mauscursors zu nutzen, wie zum Beispiel das Programm WiinRemote[1] für Windows oder das Darwiin Remote[2] für Mac OS X 10.4. Die Software GlovePie[3] bietet die Möglichkeit, mit beliebigen Eingaberäten die Eingabe von Keyboard oder Joystick zu emulieren und so Windows-Anwendungen zu bedienen. Zu den von GlovePie unterstützten Geräten gehört auch die WiiMote. Das Pinocchio-System ermöglicht das Dirigieren eines virtuellen Orchesters mit Hilfe der WiiMote [Bruegge et al., 2007].

4.1.3 WiiMote Zugriffsbibliotheken

Mittlerweile gibt es für die verschiedensten Programmiersprachen Bibliotheken, die den Zugriff vom PC auf die WiiMote ermöglichen.

[1] `http://onakasuita.org/wii/index-e.html`
[2] `http://www.wiili.org/index.php/DarwiinRemote`
[3] `http://carl.kenner.googlepages.com/glovepie`

- Java-Programmierern steht mit WiiRemoteJ[4] eine komplett dokumentierte Javabibliothek zur Verfügung, die einen Zugriff auf die WiiMote ermöglicht.
- WiimoteLib[5] ist in C# und VB.NET implementiert und ermöglicht die Verwendung der WiiMote in .NET Anwendungen.
- WiiYourself[6] ist eine C++ Library, die unter Windows umfangreichen Zugriff auf sogar mehrere WiiMotes bietet.
- Weitere Bibliotheken sind für Python, Perl, C und anderen Programmiersprachen verfügbar[7].

Die vorgestellten Bibliotheken bieten bereits beinah einen vollständigen Zugriff auf die WiiMote-Funktionalitäten. Hierzu gehört das Auslesen der aktuellen Statusinformationen wie Batterie- und Knopfzustände, Beschleunigungswerte und Infrarotsensordaten. Auch die Daten von evtl. an die WiiMote angeschlossenen Erweiterungscontroller sind zugänglich. Desweiteren können die LED-Lampen und der Vibrationseffekt an bzw. ausgeschaltet werden. Da die Anwendungen dieser Arbeit in C# programmiert sind, wird WiimoteLib für den Zugriff auf die WiiMote verwendet.

4.2 Beschleunigungsmessung

Dieser Abschnitt gibt einen technischen Überblick über die in die WiiMote eingebauten Sensoren und beschreibt die von ihnen erzeugten Signale anhand dreier Gesten.

4.2.1 Beschleunigungssensor

Die Beschleunigungsinformationen werden mittels eines linearen 3-Achsen-Beschleunigungsmessers aufgezeichnet. Hierfür ist ein ADXL330 Schaltkreis der Firma Analog Devices in die WiiMote eingebaut, der in der Lage ist, Beschleunigungen von -/+3 g zu messen, also bis zum dreifachen der Erdbeschleunigung. Eine kleine in den Schaltkreis eingebaute Feder aus Silikon ist für die Messung der Beschleunigung zuständig. Durch das Wirken einer Kraft auf die

[4] `http://www.wiili.org/WiiremoteJ`
[5] `http://www.wiili.org/index.php/WiimoteLib`
[6] `http://wiiyourself.gl.tter.org/`
[7] `http://wiili.org/index.php/Wiimote_driver`

Feder verändert sich ihre elektrische Kapazität und somit auch die elektrische Spannung, die als analoger Wert gemessen und in ein digitales Signal zur Beschreibung der Beschleunigung umgewandelt wird.

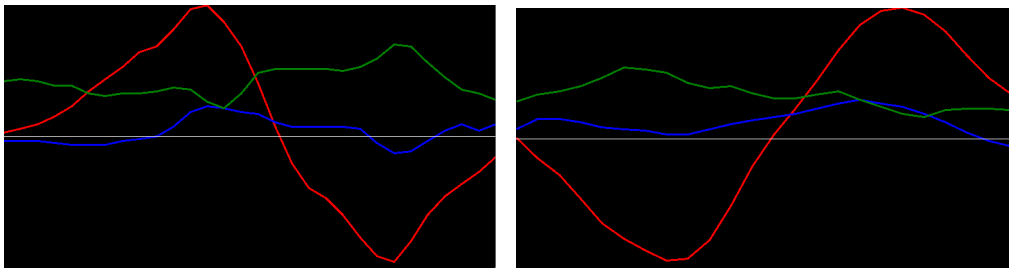
4.2.2 Koordinatensystem

Der Sensor verwendet ein rechtshändiges Koordinatensystem, dessen positive X-Achse nach links, positive Y-Achse nach vorne und positive Z-Achse nach oben gerichtet ist. Liegt die WiiMote mit der Unterseite nach unten auf dem Tisch, wird ein Wert von 1g (1g ist die Erdbeschleunigung) in Z-Richtung angezeigt. Das ist Kraft, die die Hand gegen die Erdanziehung ausüben muss, um den Controller in seiner Position zu halten. Im freien Fall hat der Betrag den Wert von 0g. Bei schiefer Haltung ohne Bewegung ergibt der Betrag des Beschleunigungsvektors 1g.

Die WiiMote übermittelt kontinuierlich den aktuellen Zustand ihrer Sensoren. Betrachtet man die Beschleunigungswerte zu den drei Achsen gegen die Zeit, so erhält man eine Beschreibung der Bewegung. Abbildungen 4.2 zeigt zwei Beispiele hierfür.

- **Linksbewegung**
  In Abbildung a) wird eine Linksbewegung beschrieben. Die rote, die X-Achse beschreibende Kurve erreicht zunächst ein Maximum und anschließend ein Minimum mit beinah gleichem Amplitudenbetrag. Dies gibt exakt die Bewegung in Linksrichtung wieder. Zunächst erfahren WiiMote und Hand eine Beschleunigung in Linksrichtung. Am Ende der Geste stoppt die Bewegung der Hand. Hierzu muss die Geschwindigkeit der Hand wieder auf null gebremst werden wofür eine ähnliche hohe Beschleunigung von gleicher Dauer in entgegengesetzte Richtung wirkt. Der Ausschlag der blauen Kurve in Y-Achsen-Richtung resultiert daher, dass die Bewegung nicht sauber ausschließlich in eine Richtung ausgeführt wurde. Der Wert der Z-Achse (grüne Kurve) pendelt um einen positiven Wert und stellt die Erdbeschleunigung dar, die konstant bei 1g liegt.

- **Rechtsbewegung**
  Betrachtet man die Rechtsbewegung in Abbildung b), erkennt man eine der Linksbewegung recht ähnliche Kurve mit dem Unterschied, dass die rote Kurve sich diesmal genau umkehrt verhält. Die Kurve erreicht zunächst ein Minimum, dann ein Maximum und beschreibt dadurch eine Beschleunigung nach rechts mit anschließender Bremsung.



(a) Linksbewegung (b) Rechtsbewegung

Abbildung 4.2: Beispiele für die Beschleunigungswerte zweier Bewegungen. Entscheidend für die Links- und Rechtsbewegung ist die rot dargestellte Sensorinformation der X-Achse. Diese verhält sich bei beiden Bewegungen genau entgegengesetzt.

4.2.3 Messen einer Bewegung

In Abbildung 4.3 werden die gemessenen Werte einer Kreisbewegung dargestellt. Die grüne Kurve beschreibt erst eine Aufwärts-, dann eine Abwärtsbewegung. Während der Aufwärtsbewegung in der ersten Hälfte der Geste erkennt man, dass die rote Linie eine komplette links-rechts Bewegung beschreibt. Die Ursache hierfür ist eine links-rechts-Bewegung, die die Hand während der ersten Hälfte des Kreises ausführt. Während der Abwärtsbewegung in der zweiten Hälfte erfolgt entgegengesetzt eine rechts-links Bewegung.

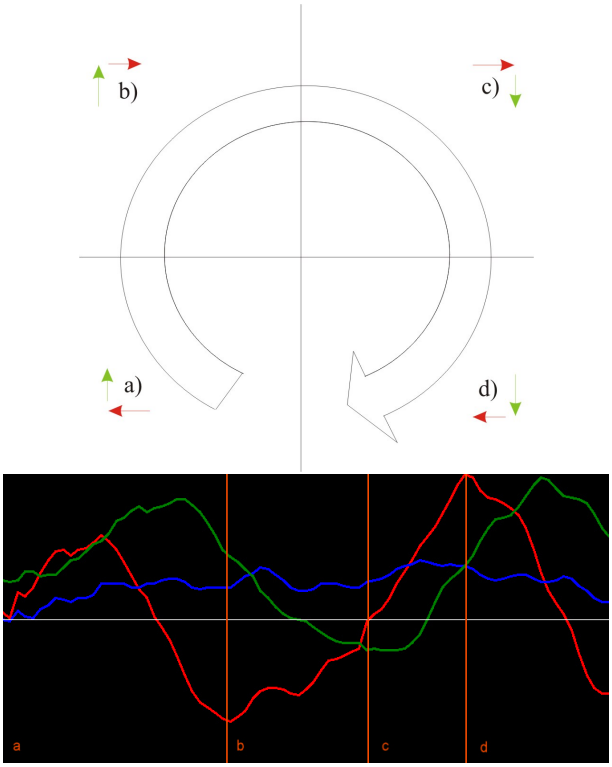


Abbildung 4.3: **a)** *Rote Kurve:* Linksbewegung mit Beschleunigung und Bremsung. *Grüne Kurve:* Beschleunigung nach oben mit Erreichen der Höchstgeschwindigkeit kurz vor Erreichen der Bewegungsphase b. **b)** *Rote Kurve:* Erreichen der Höchstgeschwindigkeit in Rechtsrichtung am Ende der Bewegungsphase. Die Beschleunigung wird dabei null. *Grüne Kurve:* Maximale negative Beschleunigung am Wendepunkt zwischen Auf- und Abbewegung am Ende der Bewegungsphase. **c)** *Rote Kurve:* Erreichen des Wendepunkts zwischen Rechts- und Linksbewegung mit maximaler Beschleunigung am Ende der Bewegungsphase. *Grüne Kurve:* Erreichen der Maximalgeschwindigkeit in Abwärstbewegung am Ende der Bewegungsphase. Die Beschleunigung liegt hier bei 1g **d)** *Rote Kurve:* Erreichen der Höchstgeschwindigkeit in Linksrichtung mit anschließender Bremsung. *Grüne Kurve:* Bremsung der Abwärtsbewegung.

4.3 Zusammenfassung

In diesem Kapitel wurde die Spielekonsole Wii vorgestellt und ein näherer Einblick in die Funktionsweise ihres Steuergerätes, die WiiMote, gegeben, die aufgrund ihres günstigen Preises und ihrer weiten Verbreitung für die Aufzeichnung von Gesten ausgewählt wurde. Die WiiMote kann Beschleunigungsdaten aufzeichnen und über Bluetooth an die Wii-Konsole senden. Es existieren Bibliotheken, die ein Einlesen der gemessenen Daten in den PC ermöglichen. Für die Arbeit wurde hierfür die gut dokumentierte und in C# implementierte Bibliothek WiimoteLib ausgewählt.

An den Beispielen von Links-,Rechts- und Kreisbewegungen wurde gezeigt, wie Gesten durch ihre Beschleunigungswerte beschrieben werden. Hierbei hat jede Geste ihre charakteristischen Merkmale. Wichtig sind vor allem die Kurven für jede Koordinatenachse und ihr Zusammenspiel miteinander. Zusätzlich von Interesse sind die Länge der Bewegung und die Intensität.

Die Gesten lassen sich anhand der charakteristischen Merkmale ihrer Beschleunigungswerte voneinander unterscheiden. Findet man geeignete, diese Merkmale beschreibende Attribute und nutzt diese als Eingabedaten für einen Klassifikator, können Entscheidungsfunktionen aufgestellt werden, die Beschleunigungswerte ihren zugehörigen Gesten automatisch zuordnen. Wie dies im Detail umgesetzt wird, beschreibt das folgende Kapitel.

Kapitel 5

Gestenklassifikation

Bei der Klassifikation von Gesten werden neu zu erkennende Gesten mit zuvor erlernten Prototypen verglichen. Die Hauptproblematik hierbei ist, dass eine Geste mit geringer Wahrscheinlichkeit zweimal identisch ausgeführt wird. Daher müssen Verfahren entwickelt werden, die kleinere Abweichungen in den Bewegungen zulassen. Hierzu werden in anderen Arbeiten zum Beispiel Verfahren aus der Spracherkennung verwendet.

5.1 Gestenerkennung in verwandten Projekten

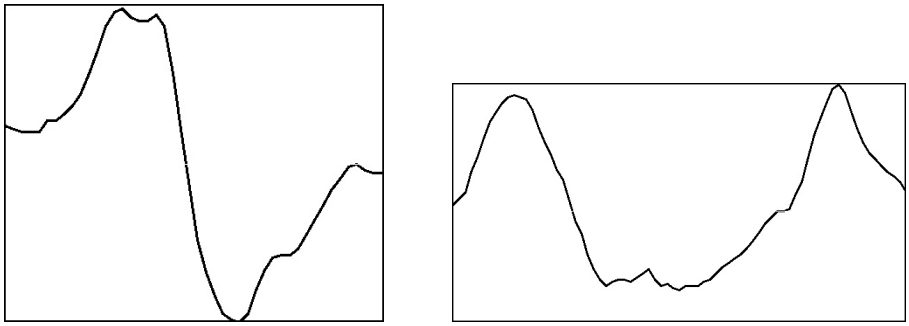
Das Erkennen von Gesten wird mittels unterschiedlichster Verfahren realisiert, unter anderem mit Neuronalen Netzen [Boehm et al., 1994] oder Neuro-Fuzzy-Systemen [Bailador and Guadarrama, 2007]. Poppinga verwendet für die Klassifizierung von WiiMote-Gesten Hidden Markov Models [Schlömer et al., 2008], die auch schon bei anderen Gestenerkennungsprojekten verwendet wurden [Mäntyjärvi et al., 2004].

5.1.1 MIT - An Inertial Measurement Framework for Gesture Recognition and Applications

Ari Y. Benbasat und Joseph A. Paradiso vom MIT [Benbasat and Paradiso, 2001] definieren atomare Gesten, die nicht weiter unterteilt werden können und aus denen sich längere Gesten zusammensetzen. Der Vorteil hierbei ist, dass nur eine geringe Anzahl an Gesten erkannt werden muss, jede andere Geste kann hieraus synthetisiert werden. Bei den Untersu-

chungen gelang es, alle Gesten in zwei atomare Bewegungen zu unterteilen, die *straight-line* - und die *there-and-back* -Bewegung (siehe Abbildung 5.1).

Die gemessenen Signale wurden auf Extremwerte untersucht und so in ihre atomaren Bewegungen zerlegt. Durch die konsekutive und konkurrierende Kombinationen der atomaren Gesten und die zusätzliche Betrachtung von Dauer und Amplitude der Bewegung, können komplexere Gesten definiert und wiedererkannt werden.



(a) Straight-Line (b) There-And-Back

Abbildung 5.1: Darstellung der Kurven der atomaren Gesten, in die eine Bewegung aufgeteilt werden kann. a) Die *straight-line*-Bewegung hat zwei Höchstwerte, die aufgrund der Beschleunigung und anschließenden Bremsung des Arms entstehen. b) Bei der *there-and-back*-Bewegung kommt eine Beschleunigung in die entgegengesetzte Richtung und eine weitere Bremsung hinzu. Da Bremsung der Bewegung in die Hinrichtung und Beschleunigung in die Rückrichtung ineinander übergehen, fallen ihre beiden Höchstwerte zusammen, weshalb die Bewegung anhand dreier Höchstwerte beschrieben wird (Quelle [Benbasat and Paradiso, 2001]).

5.1.2 XWand

Beim in Abschnitt 2.3 vorgestellten XWand-Projekt [Wilsona and Wilson, 2004] wurde versucht, Ansätze aus der Spracherkennung zu verwenden. Es wurden drei Algorithmen implementiert und getestet.

- **Linear Time Warping (LTW)**
  Es werden mehrere Prototypen der Gesten aufgenommen. Für den Vergleich werden die Signale in Dauer und Amplitude normalisiert und der Unterschied zwischen Aufnahme und Prototyp über die quadratische euklidische Distanz ermittelt. Der Prototyp mit der geringsten Distanz wird als erkannte Geste ausgewählt.

- **Dynamic Time Warping (DTW)**
  Dieser Ansatz ist ähnlich dem LTW nur dass nichtlineare Zeitdifferenz besser berücksichtigt werden, die auftreten wenn man Gesten mit unterschiedlichen Geschwindigkeiten ausführt. Hierbei werden Signale unterschiedlicher Länge mit Hilfe dynamischer Programmierung optimal aufeinander ausgerichtet und per quadratischer euklidischer Distanz miteinander verglichen.

- **Hidden Markov Model (HMM)**
  Der Hidden Markov Erkenner wurde ursprünglich für die Spracherkennung entwickelt und liefert ein Wahrscheinlichkeitsmodell, das dynamische, zeitvariante Gesten trainieren und wiedererkennen kann. Für das XWand-Projekt hat man das HMM-Framework HTK Toolkit 3.0 verwendet.

Während der Evaluation wurden sechs Probanden getestet, die jeweils 42 Gesten ausführten. Aus einem Wortschatz von sieben Gesten, bestehend aus Auf-, Ab-, Links- und Rechtsbewegung, Bewegung im und gegen den Uhrzeigersinn und einer Geste, die das Aufnehmen eines Objekts darstellt, sollten die drei Verfahren die korrekte Geste erkennen. Die Ergebnisse waren mittelmäßig bis sehr gut. Während LTW eine Genauigkeit von nur 42% ermittelte, war der DTW Algorithmus mit 72% schon um einiges effizienter. Das komplexere Verfahren mit dem Hidden Markov Model ergab sogar eine Genauigkeit von 90%.

5.1.3 Gestenerkennung mit Klassifikatoren

Baier, Mosenlechner und Kranz verwendeten für die Klassifikation von Gesten Maschinenlernverfahren, darunter einen Aufbau mit einem hierarchisch strukturierten Neuronalen Netz, basierend auf Selbstorganisierenden Karten (SOM) und eine Support Vector Machine [Baier et al., 2007]. Ersteres Verfahren hatte eine Erkennungsrate von 80%, die SVM erkannt 76% der Gesten richtig.

5.2 Merkmalsextraktion

In dieser Arbeit wurde ein Ansatz entwickelt, der Klassifikatoren für die Erkennung von Gesten verwendet. Für die Umsetzung wurde die Maschinenlernbibliothek Weka ausgewählt, die die gängisten Maschinenlernalgorithmen implementiert (siehe Abschnitt 3.4). Aus mehreren Gründen müssen hierzu die Sensordaten einer Merkmalsextraktion unterzogen werden.

5.2.1 Problemstellung

Es gibt eine unendlich große Anzahl an Bewegungen, die in Länge, Pfad und Geschwindigkeit variieren. Selbst wenn zwei Bewegungen eine identische Geste ausdrücken sollen, so gibt es immer Abweichungen in der Ausführung, sei es dass die Bewegung schneller ausgeführt wurde oder dass die Richtung der Bewegungspfade leicht voneinander abweichen. Bei den Beschleunigungsdaten machen sich diese Unterschiede durch unterschiedliche Werte und eine unterschiedliche Menge an Messdaten bemerkbar.

Ein erstes Problem das sich hieraus ergibt ist, dass für den Vergleich unterschiedlich langer Bewegungen die Information fehlt, welche Werte miteinander zu vergleichen sind. Als Beispiel betrachte man zwei Kreisbewegungen, eine doppelt so schnell ausgeführt wie die andere. Da die WiiMote pro Sekunde eine Konstante Anzahl an Messwerten verschickt, liegen zum ersten Kreis gegenüber dem zweiten doppelt so viele Werte vor. Vergleicht man stur die Werte, so würde die komplette kurze Kreisbewegung ausschließlich mit der ersten Hälfte der Werte der langsameren Bewegung verglichen. Eine Lösung wäre, nur jeden zweiten Wert der langsamen Bewegung zu verwenden. Jedoch liegt in den meisten Fällen nicht genau eine doppelte Anzahl an Messwerten vor.

Die Verwendung von Weka für die Klassifikation erweitert die Problematik insofern, dass Weka die zu klassifizierenden Objekte durch eine fixe Anzahl an Attributen beschreibt. Daher muss für alle Gesten, unabhängig von ihrer Länge, eine Vorverarbeitung durch Merkmalsextraktion durchgeführt werden.

Folgende Anforderungen werden hierbei gestellt:

- Feste Anzahl an Attributen
- Möglichst charakteristische Beschreibung der Messwerte
- Toleranz gegenüber kleinen Abweichungen zwischen gleichartigen Gesten
- Geringe Anzahl an Attributen, um die Rechenzeit der Klassifikationsalgorithmen gering zu halten

Für diese Arbeit wurden zwei unterschiedlichen Verfahren zur Merkmalsextraktion entwickelt, die im Folgenden vorgestellt werden.

5.2.2 Äquidistantes Gitter

Idee

Im ersten Ansatz wurde sich der Problemstellung gewidmet, eine feste Anzahl an Merkmalen zu finden. Dies wird gelöst, indem die Signale so gestaucht oder gestreckt werden, dass sie gleich lang werden. Um diesen Effekt zu erreichen, legen wir eine feste Anzahl an Abtastpunkten fest und verteilen diese äquidistant über die komplette Messung.

Vergleichen kann man dieses Verfahren mit dem Digitalisieren von analogen Signalen. Hierbei wird eine Abtastrate festgelegt und die Werte des stetigen Signals in einer konstanten Frequenz ausgelesen. Beim Erkennen von Gesten sollte die Abtastrate jedoch nicht fest sein, da sonst bei unterschiedlich langen Gesten die Anzahl an Werten variieren würde. Daher wird die Abtastrate an die Länge der Geste angepasst. Man stelle sich ein Gitter mit einer festen Anzahl an Abtastpunkten vor. Dieses wird so gestreckt oder gestaucht, bis es die Länge des Signals erreicht (siehe Abbildung 5.2).

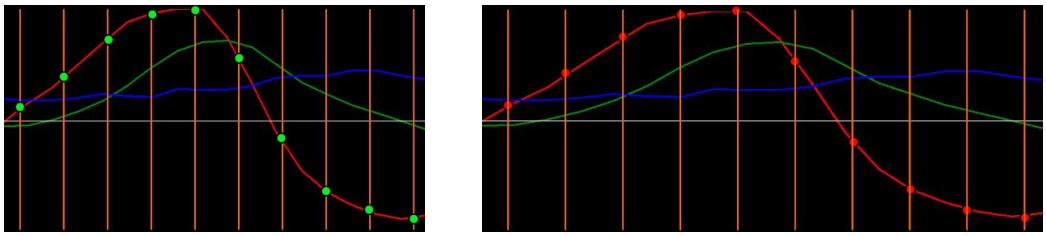


Abbildung 5.2: Je nach Signallänge werden die Abstände der Abtastpunkte angepasst. Die Anzahl der Punkte bleibt somit gleich.

Gehen wir davon aus, dass man n Abtastpunkte hat und die Messwerte diskret auf dem Vektor v der Größe m liegen. Aufgabe ist, die n Abtastpunkte auf das Intervall $[0, m-1]$ zu verteilen. Für den i-ten Abtastpunkt p_i mit $i \in \{0, 1, ..., n-1\}$ ergibt dies:

$$p_i = \frac{(m-1) \cdot i}{n-1}$$

Im Gegensatz zu einem stetigen analogen Signal sind die Messwerte der Beschleunigungssensoren diskret. Daher kann man nicht automatisch davon ausgehen, dass ein Abtastpunkt genau auf einem Messwert liegt, sondern im häufigsten Fall zwischen zwei Werten. Um möglichst genaue Werte zu erhalten, wird mit linearer Interpolation gearbeitet, die die beiden Werte in der Umgebung des Abtastpunkt mit einbezieht. Sei a die nächst kleinere ganze Zahl zu p_i und b die nächst größere ganze Zahl und v_a bzw. v_b die Werte des diskreten

Signals an den Indizes a und b. Dann ist der Wert w_i am Abtastpunkt p_i:

$$w_i = v_a \cdot (1 - (p_i - a)) + v_b \cdot (p_i - a)$$

Die Werte der Abtastpunkte werden zu allen drei Kurven ermittelt und anschließend aneinandergereiht, so dass man bei 16 Abtastpunkten 48 Attribute zur Beschreibung der Kurve erhält. Um die Länge der Bewegung nicht komplett außer Acht zu lassen, wird die Anzahl der gemessenen Werte ebenfalls als Attribut verwendet, so dass man insgesamt auf eine Zahl von 49 Attributen kommt.

Vor- und Nachteile

Während der Evaluation (s. Abschnitt 7.3.1) erreichte die Präzision des Äquidistanten Gitterverfahrens im besten Fall (mit Support Vector Machine) 73%. Angelehnt an den LTW-Algorithmus wurde zusätzlich ein Versuch durchgeführt, bei dem die Messwerte vor der Merkmalsextraktion auf das Intervall [0,1] normalisiert werden. Hierbei lag die Präzision ebenfalls bei 72%.

Schwächen zeigt das Gittermodell bei periodischen Bewegungen. Periodische Bewegungen setzen sich aus Wiederholungen einer kurzen Bewegung zusammen. Hierbei soll die Anzahl der Wiederholungen für das Wiedererkennen der Gesten nicht von Bedeutung sein. Will man zum Beispiel die Geste *Winken* beschreiben, ist nicht entscheidend ob zwei- oder fünfmal gewunken wird, beide Bewegungen sollen als Winken klassifiziert werden. Legt man eine Kurve mit fünf Perioden (s. Abbildung 5.3) und eine mit zweien übereinander, so wird schnell deutlich, dass man die Werte schlecht direkt miteinander vergleichen kann. Das Gittermodell verwendet jedoch einen Ansatz des direkten Vergleichs und liefert daher für periodische Bewegungen schlechte Ergebnisse.

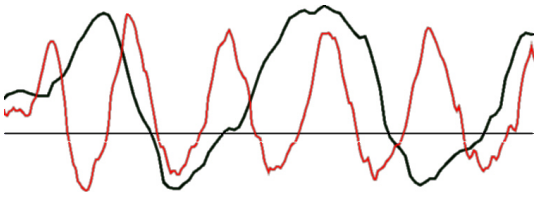


Abbildung 5.3: Bei periodischen Bewegungen mit einer unterschiedlichen Anzahl an Wiederholungen fallen nur sehr wenige Punkte aufeinander.

5.2.3 Attributbeschreibendes Modell

Idee

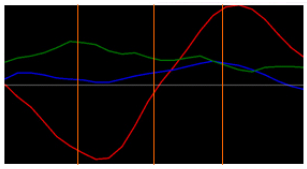
Als zweiter Ansatz wurde ein Modell entwickelt, das die Gesten anhand mathematischer Eigenschaften der drei Kurven beschreibt. Es werden lokale Eigenschaften, wie z.B. Extremstellen an bestimmten Punkten der Kurve und globale Eigenschaften wie Länge oder Mittelwert berücksichtigt. Ziel herbei ist, die Geste anhand ihrer speziellen Merkmale wiederzuerkennen.

Verwendete Attribute

Attribute, die für jede der drei Kurven getrennt ermittelt werden:

- Maximalwert / Minimalwert: Maximal- und Minimalwert geben an, mit welcher Intensität die Bewegung ausgeführt wird und welche Bewegungsdimension besonders dominant ist.
- Mittelwert / Standardabweichung: Dienen der allgemeinen Beschreibung der Kurve. Anhand des Mittelwertes wird beschrieben, um welchen Wert sich die Kurve bewegt. Die Standardabweichung gibt an, wie volatil die Kurve ist.
- Anzahl an lokalen Maximal- und Minimalstellen in bestimmten Intervallen: Höchstwerte sind sehr charakteristisch für eine Bewegung und beschreiben die Kurve in bestimmten Intervallen der Bewegung. Im nächsten Abschnitt wird genauer beschrieben, wie sie ermittelt werden.
- Anfangs- und Endzustand: Anfangs- und Endzustand des Controllers können sehr aussagekräftig für eine Bewegung sein. Will man zum Beispiel die Bewegung Tennisaufschlag von der Bewegung Vorhand unterscheiden, so ist die Anfangshaltung des Controllers grundlegend verschieden. Im ersten Fall wird er senkrecht über dem Kopf gehalten, im anderen Fall seitlich vom Körper weg. Jede der beiden Haltungen wird über 3 Werte (für die 3 Dimensionen) beschrieben.

Hinzu kommt die Länge der Geste als wichtige charakteristische Eigenschaft zur Unterscheidung von langen, komplexen und kurzen, einfachen Bewegungen. Zählt man alle Attribute zusammen erhält man im Ganzen für eine Geste 43 Werte (3 Maximalwerte + 3 Minimalwerte + 3 Mittelwerte + 3 Standardabweichungen + 24 für die Anzahl lokaler Extremstellen + 6 Werte für Anfangs und Endhaltung + 1 Länge).



Attribut	x	y	z
Mittelwert	0,135	0,407	1,046
Standardabweichung	2,01	0,28	0,32
Maximalwert	3,08	0,92	1,68
Minimalwert	-2,84	-0,16	0,52
Lokale Maxima	0 0 0 1	0 0 0 0	0 0 0 0
Lokale Minima	0 1 0 0	0 0 0 0	0 0 0 0
Bewegungslänge	24		
Startzustand	0,04	0,24	0,88
Endzustand	1,08	-0,16	0,68

Abbildung 5.4: Die Tabelle listet die Werte auf, die für obige Kurve beim Attributbeschreibenden Modell ermittelt werden. Die Anzahl der lokalen Extremstellen wird für die vier abgebildeten Intervalle getrennt ermittelt.

Algorithmus zum Ermitteln der lokalen Extremstellen

Im Folgenden wird der Algorithmus beschrieben, mit dem lokale Extremstellen ermittelt werden. Da die Kurven diskret vorliegen, kann nicht mit Methoden aus der Analysis gearbeitet werden. Deswegen wird im ersten Schritt die Umgebung jedes Messpunktes untersucht. Sei ϵ die Variable, die die Größe der zu untersuchenden Umgebung angibt. Wählt man $\epsilon = 6$, werden sechs Werte links und sechs Werte rechts vom aktuellen Punkt betrachtet. Enthält die Umgebung keinen Punkt der größer bzw. kleiner ist als der zu untersuchende Punkt, so hat man ein potentielles Maximum bzw. Minimum. Potentiell deswegen, weil Extremstellen mit nur geringer Amplitude sich nicht sehr gut eignen, um die Charakteristik einer Kurve zu beschreiben. Sie können auch durch kleine Wackler, Zittern oder Rauschen entstanden sein, die über die Bewegung selber nichts aussagen und daher nicht berücksichtigt werden sollen.

Um diese Extremstellen zu filtern wird eine Heuristik verwendet. Idee dieser Heuristik ist, nur Extremwerte zu verwenden, die prägnant für eine Kurve sind. Hierzu wird die höchste Standardabweichung der drei Kurven herangezogen. Diese dient zur Ermittlung eines Grenzwertes, der bestimmt, ob ein Wert als Extremwert gezählt wird. Hierzu muss der Abstand des Extremwertes vom Mittelwert einer Kurve größer sein, als die Hälfte der höchsten Standardabweichung.

Ist σ_{max} die größte Standardabweichung, μ der Mittelwert und x_{max} der Wert der möglichen Extremstelle, muss für ein Maximum gelten:

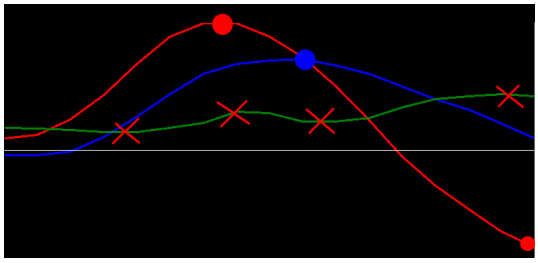


Abbildung 5.5: Die Grafik beschreibt, welche der Extremstellen charakteristisch für die Beschreibung der Geste sind. Die mit Punkten markierten Stellen werden gezählt. Mit einem Kreuz markierte Werte weichen zu wenig von dem Mittelwert der Kurve ab und werden nicht berücksichtigt

$$x_{max} > \mu + \frac{\sigma_{max}}{2}$$

Für ein Minimum gilt dementsprechend:

$$x_{min} < \mu - \frac{\sigma_{max}}{2}$$

Das Erfassen der Extremstellen erfolgt quantitativ und wird zwischen Minimal- und Maximalstellen getrennt durchgeführt. Hierzu wird die Bewegung in vier Intervalle unterteilt und zu jedem Intervall jeweils die Anzahl der vorkommenden Maximal- und Minimalwerte gezählt. Bei den drei Kurven kommt man so auf 12 Werte, verwendet also insgesamt für die Beschreibung der Extremstellen 24 Attribute.

Vor- und Nachteile

Die Evaluation ergab, dass bei dem Attributbeschreibenden Modell mit einer Präzision von im besten Fall 65% weniger Gesten korrekt erkannt wurden, als beim Gitteransatz (siehe Evaluation Kapitel 7.3.1). Eindeutige Stärken des Modells liegen bei periodischen Bewegungen, für die das Gittermodell nicht geeignet ist. Das Attributbeschreibende Modell hat hier aufgrund der Tatsache, dass Mittelwert, Varianz, Maximal- und Minimalwert unabhängig von der Anzahl an Wiederholungen ähnlich bleiben, die Möglichkeit, Geste wiederzuerkennen, auch wenn Unterschiede bei der Länge oder der Anzahl an lokalen Extremwerte existieren.

5.3 Zusammenfassung

Dieses Kapitel beschäftigte sich mit der Frage, wie die Werte der Beschleunigungssensoren vorverarbeitet werden müssen, damit sie miteinander verglichen werden können. Andere Projekte verwenden hierfür Verfahren aus der Spracherkennung oder haben eigene Algorithmen entwickelt, die Kurven anhand ihrer Höchstwerte wiedererkennen. In Weka, der Bibliothek die für die Klassifikation verwendet wird, werden Modelle anhand einer festen Anzahl an Attributen beschrieben. Problem hierbei war, dass die Anzahl der zu verschiedenen Bewegungen gemessenen Werte unterschiedlich ist. Daher mussten Extraktionsverfahren zur Ermittlung einer festen Anzahl an Merkmalen entwickelt werden, die die Kurven möglichst gut beschreiben. In diesem Kapitel wurden zwei selbst entwickelte Verfahren vorgestellt.

- **Das Äquidistante Gitter**
  Man legt ein Gitter mit einer festen Anzahl an Abtastpunkten über die gemessenen Werte einer Bewegung. Der Gitterabstand bleibt hierbei konstant. Die Werte der Kurve an diesen Punkten werden durch lineare Interpolation der beiden Umgebungswerte ermittelt.

- **Attributbeschreibendes Modell**
  Die Kurven werden durch mathematische Eigenschaften wie Mittelwert, Standardabweichung oder Extremstellen in bestimmten Intervallen untersucht.

Beide Verfahren wurden in das Toolkit TaKG (Kapitel 6) implementiert und während der Evaluation (Kapitel 7) auf ihre Präzision getestet. Das Äquidistante Gitter lieferte hierbei die besseren Ergebnisse. Die Stärke des Attributbeschreibendem Verfahren liegt in der Erkennung von periodischen Bewegungen mit einer nicht festgelegten Anzahl an Wiederholungen.

Kapitel 6

TaKG - Toolkit zur automatischen Klassifikation von Gesten

TaKG ist ein **T**oolkit zur **a**utomatischen **K**lassifikation von **G**esten, das die Integration gestengesteuerter Interaktionen in Anwendungen vereinfachen soll. Das Hauptaugenmerk liegt hierbei auf einfach zu implementierenden Schnittstellen, die die Funktionalitäten des Toolkits ohne großes Hintergrundwissen nutzbar machen sollen.

Ein weiterer wichtiger Punkt ist, das Toolkit möglichst dynamisch zu gestalten, so dass einzelne Komponenten schnell ausgetauscht werden können. Hierfür ist es notwendig, den Aufbau der Software stark zu modularisieren. Betrachtet man zum Beispiel die Erfassung der Sensordaten, so wird diese in einem eigenen Modul realisiert, für das feste Schnittstellen definiert sind. Tauscht man den Sensor aus, müssen diese Schnittstellen in dem für den neuen Sensor geschriebenen Code zur Verfügung stehen. Die sensorspezifischen Aufgaben werden in dem sich hinter den Schnittstellen verbergenden Teil realisiert. Für die WiiMote werden hier mehrere Aufgaben erfüllt. Zunächst muss auf unterster Ebene die Bluetoothkommunikation realisiert und das WiiMote initialisiert werden. Die anschließend empfangenen Daten müssen ausgelesen und richtig gedeutet werden, so dass man am Ende einen die Geste beschreibenden Datensatz erhält.

6.1 Übersicht über die Architekturebenen

Das Toolkit ist grob in zwei Ebenen unterteilt, die Klassifikations- und die Anwendungsebene (s. Abbildung 6.1). In der Anwendungsebene sind Module

enthalten, die das Erzeugen einer gestengesteuerten Anwendung unterstützen. Dazu gehört ein Modul, das die Bewegungsdaten von einem Sensor einliest und in ein passendes Format bringt, um sie anschließend weiterzuverarbeiten. Ein anderes Modul regelt die Kommunikation mit der Klassifikationsebene, stellt Lern- und Klassifikationsanfragen an diese durch und empfängt deren Antworten.

Die Klassifikationsebene übernimmt die Aufgaben, die die Klassifikation der Daten betreffen, so wie Vorverarbeitung der Messungen und Aufrufen der Klassifikationsalgorithmen. Weiterhin kommuniziert sie mit der Anwendungsebene.

Die beiden Ebenen werden voneinander getrennt, so dass jede ihren eigenen Prozessablauf hat und sie sogar auf getrennten Systemen laufen können. Die Kommunikation erfolgt über eine Socketverbindung im TCP-Protokoll. Als Kommunikationssprache dient eine Syntax im XML-Format, die die Anfragen und Antworten beschreibt.

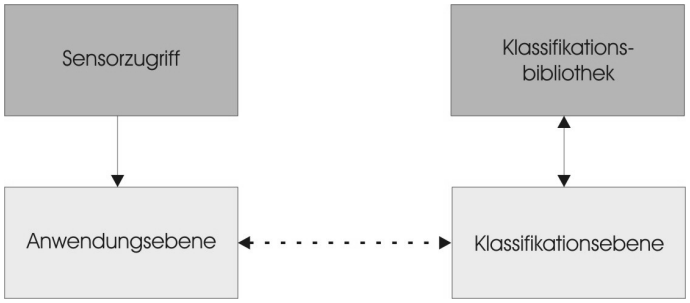


Abbildung 6.1: Der Aufbau des TaKG-Toolkits ist grob in zwei Hauptebenen unterteilt. Die Anwendungsebene greift auf Sensordaten zu und schickt Aufträge an die Klassifikationsebene. Diese behandelt die Aufträge, erlernt und erkennt Gesten mit Hilfe von externen Klassifikatoren und gibt der Anwendungsebene Antwort auf ihre Anfragen zurück.

6.2 Klassifikationsebene

6.2.1 Aufgaben

Die Klassifikationsebene (s. Abbildung 6.2) ist für das Erlernen und Klassifizieren der Gesten zuständig. Hier werden Lern- und Klassifikationsanfragen empfangen. Die in den Anfragen enthaltenen Signaldaten werden zunächst ausgelesen. Während der Vorverarbeitung werden die Merkmale der Signale extrahiert und diese anschließend, je nach Anfragetyp, entweder als Trainingsdaten verwendet oder klassifiziert. Resultate der zu den Anfragen ausgeführten Vorgänge

werden zurück an den Anfragesteller geschickt. Nebenher werden Datenstrukturen angelegt, die die bisher erlernten Gesten speichern und so für jeden Benutzer einen Trainingsdatensatz anlegen, auf den bei Bedarf zurückgegriffen werden kann. Die Klassifikationsebene ist in Java implementiert und arbeitet als Daemon, läuft also benutzerunabhängig im Hintergrund und soll anfragenden Anwendungen Klassifikationsdienste zur Verfügung stellen.

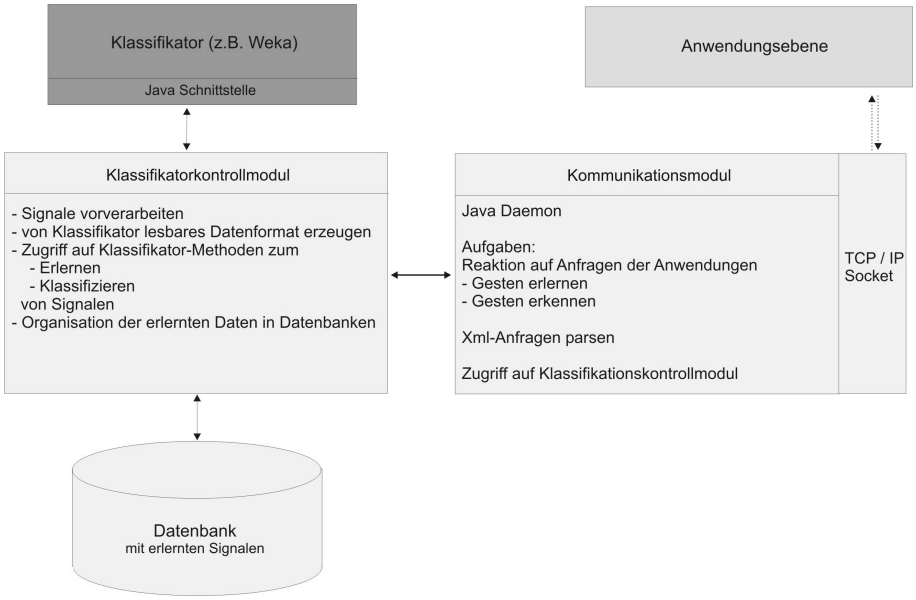


Abbildung 6.2: Übersicht über die Module der Klassifikationsebene.

6.2.2 Kommunikationsmodul

Der Datenverkehr wird durch ein Modul geregelt, das die Schnittstelle zur Anwendungsebene bildet. Die Schnittstelle selbst ist eine Serveranwendung, die über einen Port ansprechbar ist. Diese empfängt XML-Anfragen, bestehend aus einem Befehlsteil, der die Art der aktuellen Anfrage beschreibt, und häufig aus einem Datenteil, in dem das zu bearbeitende Signal enthalten ist (eine genauere Beschreibung der XML-Strukuren ist in Abschnitt 6.5 nachzulesen). Sobald eine Anfrage empfangen wird, liest ein XML-Parser den Befehlsteil und die Signale aus der Anfrage aus und ruft die entsprechende Methode des Klassifikatorkontrollmoduls auf. Die auf den Aufruf zurückerhaltene Antwort wird in eine XML-Struktur gepackt und an den Anfragesteller zurückgeschickt.

6.2.3 Klassifikatorkontrollmodul

Die Hauptaufgaben des Klassifikatorkontrollmoduls sind im größten Teil organisatorischer Art. Es wird klar durch seine Schnittstelle definiert und soll ausgetauscht werden können, falls ein anderer Klassifikator verwendet wird, wobei der Klassifikator eine bereits existierende Programmbibliothek wie z.B. Weka oder ein selbst implementierter Algorithmus sein kann.

Zu den Aufgaben des Klassifikatorkontrollmoduls gehört zunächst das Vorverarbeiten der einkommenden Daten. Je nach verwendetem Klassifikator sieht dies unterschiedlich aus, z.B. benötigt man für Weka eine feste Anzahl an Merkmalen, die man mit speziellen Algorithmen extrahiert (siehe Kapitel 5).

Eine weitere Aufgabe ist die Verwaltung der Benutzer und der zu ihnen erlernten Trainingsdaten. So soll das Modul feststellen können, ob bereits Daten zu einem Benutzer vorhanden sind oder ob dieser Benutzer bereits bestimmte Gesten erlernt hat. Auch müssen neue Datensätze zu Benutzern angelegt, diese erweitert, bearbeitet oder auch gelöscht werden können. Bei einkommenden Klassifikationsanfragen werden die zum Benutzer gehörenden Trainingsdaten geladen und für die Klassifikation der Geste verwendet.

Weiterhin kontrolliert das Modul den Zugriff auf die Bibliotheken der Klassifikatoren, die die Daten erlernen und klassifizieren sollen. Im Fall von Weka geht dies über die Weka API, die als Javabibliothek zur Verfügung steht.

Das Klassifikatorkontrollmodul kann ausgetauscht werden, um andere Klassifikationsalgorithmen oder -bibliotheken zu verwenden. Die Schnittstelle hierfür ist wie folgt definiert:

```
TaKGClassifier {
    public boolean commitData();
    \\ speichere Daten in der Datenbank

    public void learnData(String user, String tag,
        ArrayList<ArrayList<Double>> signals) throws IOException;
    \\ Fügt Geste in signals für Benutzer user zu Klasse tag hinzu

    public Map<String,Double> classifyData(String user,
        ArrayList<ArrayList<Double>> signals ) throws Exception;
    \\ Klassifiziert Geste in signals für Benutzer user

    public boolean checkForTag(String user, String tag);
    \\ Prüft ob erlernte Geste tag für Benutzer user vorhanden ist

    public boolean checkForUser(String user);
```

```
        \\ Prüft ob für Benutzer user erlernte Gesten vorhanden sind

        public boolean deleteUser(String user);
        \\ Löscht erlernte Gesten für Benutzer user


        public boolean deleteTag(String user, String tag);
        \\ Löscht Geste tag für Benutzer user
    }
```

Der Parameter `signals` enthält die Werte der zu erlernenden oder klassifizierenden Geste. Er ist ein Array der aus Arrays besteht, die Werte vom Typ `double` enthalten. Die einzelnen Arrays beschreiben jeweils die Kurve eines Sensors.

6.3 Anwendungsebene

6.3.1 Aufgaben

In der Anwendungsebene (siehe Grafik 6.3) werden Module zusammengefasst, deren Funktionalitäten bei der Verwendung gestengesteuerter Interaktionen regelmäßig gebraucht werden. Hierdurch wird die Entwicklung gestengesteuerter Anwendungen vereinfacht, da grundlegende Konzepte bereits vorliegen und nur noch eingebunden werden müssen. Die aktuelle Implementierung ist in C# realisiert.

6.3.2 Repräsentation von Gesten

Gesten werden mit Hilfe der Klasse `Gesture` beschrieben, die die Werte einer aufgezeichneten Geste enthält und zusätzliche Funktionen zur Verfügung stellt, wie z.B. das Zeichnen der Signalkurven in ein Bitmap oder das Auslesen und Erzeugen der XML-Repräsentation einer Geste. Die Klasse ist wie folgt definiert:

```
public class Gesture {
    public void addValues(double x, double y, double z);
    // fügt einen weiteren Abtastpunkt hinzu

    public int Count();
    // gibt die Anzahl der Messwerte zurück

    public Bitmap drawCurves(int width, int height);
    // zeichnet die Kurve des Signals in ein Bitmap und gibt dieses zurück
```

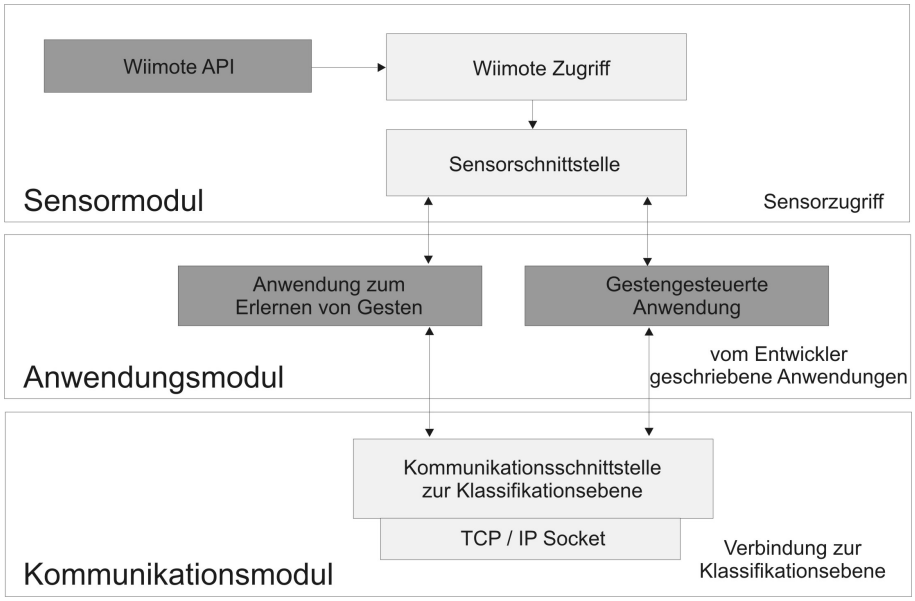


Abbildung 6.3: Übersicht über die Module der Anwendungsebene.

```
    public String createXMLRepresentation();
    // erzeugt XML-Repräsentation

    public static Gesture parseXMLRepresentation(String xmlstring);
    // liest XML-Repräsentation aus dem String xmlstring aus
}
```

6.3.3 Sensormodul

Das Sensormodul organisiert den Zugriff auf die Sensoren verschiedener Eingabegeräte. Über die Sensorenschnittstelle wird definiert, welche Methoden bei der Implementierung für die unterschiedlichen Geräte zur Verfügung stehen müssen. Auf diese Weise kann schnell ein neues Gerät integriert werden, ohne dass die Anwendung selbst bearbeitet werden muss. Will man zum Beispiel von der Steuerung mit der WiiMote auf die Steuerung mit einem Handschuh mit Beschleunigungssensoren wechseln, genügt es, den Code für den WiiMote-Zugriff zu ersetzen, vorausgesetzt man hält sich an die Vorlagen der Sensorschnittstelle.

Die Sensorschnittstelle enthält Methoden zum Starten und Stoppen der Gestenaufzeichnung. Sobald eine Geste aufgezeichnet wurde, wird diese über ein Event als `Gesture`-Instanz an die Anwendung übergeben. Die Sensorschnittstelle ist wie folgt definiert:

```
public delegate void onRecordHandler(Gesture measurement);

public class GestureRecorder {

    public event onRecordHandler onRecord;
    //wird ausgelöst, wenn eine Geste aufgezeichnet wurde
    //übergibt ein Gesture-Objekt

    public void startRecord();
    // startet die Gestenaufzeichnung

    public void stopRecord()
    // beendet die Gestenaufzeichnung
}
```

6.3.4 Anwendungsmodul

Es sind zwei Typen gestengesteuerter Anwendungen vorstellbar. Die einen beschäftigen sich mit dem Erlernen von Gesten und stellen dafür entsprechende Oberflächen zur Verfügung, die den Benutzer durch den Lernprozess führen und ihm so die Möglichkeit geben, aufgezeichnete Gesten mit Klassennamen zu versehen. Die anderen Anwendungen sollen durch die erlernten Gesten gesteuert werden können. Natürlich ist auch eine Vermischung beider Typen vorstellbar, zum Beispiel eine Anwendung, die zunächst die nötigen Gesten aufzeichnet und direkt mit den für die Interaktion benötigten Namen versieht, um sie anschließend als Eingabe zu verwenden.

Für die Modularisierung der Anwendungsebene wird zwischen den beiden Anwendungstypen nicht unterschieden. Beides sind Anwendungen, die auf Sensordaten zugreifen und diese an die Klassifikationsebene weiterleiten, um mit deren Ergebnissen zu arbeiten. Dafür werden dem Anwendungsentwickler das Sensormodul und das Kommunikationsmodul zur Verfügung gestellt.

6.3.5 Kommunikationsmodul

Das Kommunikationsmodul regelt die Kommunikation mit der Klassifikationsebene und ermöglicht über die zur Verfügung gestellten Methoden einen einfachen Zugriff auf die Gestenklassifikation. Es wird zunächst eine Verbindung zu dem Server der Klassifikationsebene aufgebaut und anschließend die Kommunikation zwischen Anwendungs- und Klassifikationsebene geregelt.

Hierzu wird die zu den Anfragen passende XML-Syntax erzeugt und an den Server geschickt. Nach Erhalt der Antwort, die ebenfalls in XML vorliegt, werden die nötigen Informationen aus der Antwort ausgelesen und an die aufrufende Anwendung zurückgegeben. Folgende Methoden werden vom Kommunikationsmodul zur Verfügung gestellt:

```
public class Classifier {
    public void Start();
    // baut die Verbindung zum Server auf

    public void Stop();
    // beendet die Verbindung zum Server

    public String classifyData(Gesture m, String user);
    // schickt eine Klassifikationsanfrage der Geste m
    // für den Benutzer user an den Server

    public bool learnData(Gesture m, String user, String tag);
    // schickt eine Lernanfrage der Geste m für den Benutzer user
    // und dem Klassennamen tag an den Server

    public bool checkForUser(String user);
    // überprüft ob Daten für den Benutzer user vorhanden sind

    public bool checkForTag(String user, String tag);
    // überprüft ob die Klasse tag für den Benutzer user erlernt wurde

    public bool deleteUser(String user);
    // löscht den Benutzer user und alle für ihn erlernte Gesten

    public bool deleteTag(String user, String tag);
    // löscht die erlernten Gesten tag für den Benutzer user

    public bool commitData();
    // übernimmt die erlernten Gesten in die Datenbank
}
```

6.4 Datenfluss

In Abbildung 6.4 ist der Datenfluss innerhalb des TaKG Toolkits abgebildet. Die gemessenen Sensorwerte werden zunächst in der Sensorschnittstelle durch ein Gesten-Objekt beschrieben. Die Anwendung nimmt das Gestenobjekt auf und stellt über die Kommunikationsschnittstelle eine XML-Anfrage an die Klassifi-

kationsebene. Die Klassifikationsebenenschnittstelle liest die Informationen aus der XML-Anfrage aus und ruft mit den darin enthaltenen Sensorwerten eine für die Anfrage geeignete Methode der Klassifikatorschnittstelle auf. Dort erfolgt die Merkmalsextraktion, deren Ergebnis als Eingabe für den Klassifikator dient. Der Klassifikator stellt eine Hypothese zu der Geste auf, die über die verschiedenen Stationen zurück zur Anwendungsebene gelangt. Dort werden die enthaltenen Informationen innerhalb der Anwendung weiterverarbeitet, zum Beispiel durch eine der Geste entsprechenden Reaktion der Anwendung.

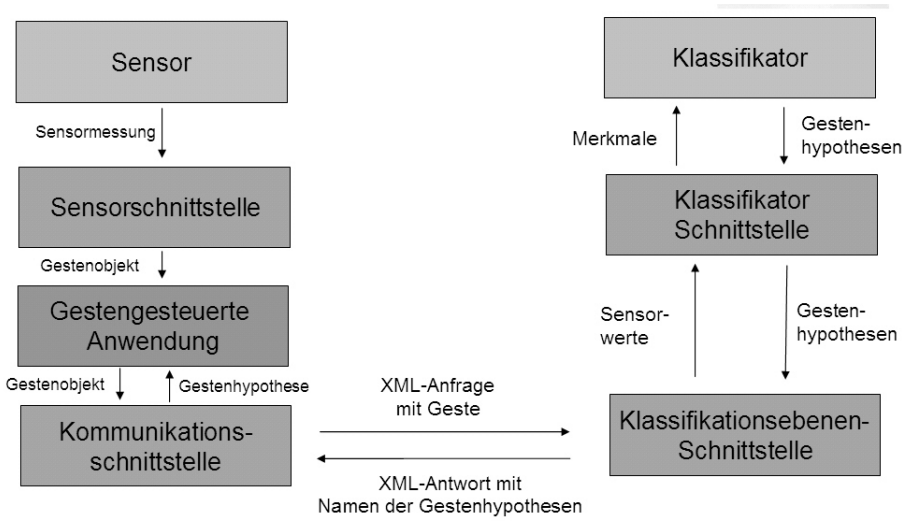


Abbildung 6.4: Datenfluss im TaKG-Toolkit

6.5 XML-Syntax der Gestenanfragen

Anfragen an den Klassifikationsserver werden in XML übermittelt. Dabei werden sie vom Tag `GestureClassificatorRequest` eingeschlossen. Die Antwort vom Server wird dementsprechend mit dem Tag `GestureClassificatorAnswer` markiert. Hier ein Beispiel für eine Anfrage:

```
<GestureRecognitionRequest>
    <LearnData user="agassi" tag="aufschlag">
        <Signal>
            <Value> 1.23</Value>
            <Value> 0.43</Value>
            ...
        </Signal>
        <Signal>
            <Value> 0.72</Value>
```

```
            <Value> 0.33</Value>
            ...
        </Signal>
        <Signal>
            <Value> 2.23</Value>
            <Value> 0.99</Value>
            ...
        </Signal>
</LearnData>
</GestureRecognitionRequest>
```

Die verschiedenen Signale einer Geste werden getrennt per `Signal`-Tag übermittelt. Die einzelnen Werte eines Signals werden nacheinander durch Verwendung des `Value`-Tags aufgelistet. Weitere nötige Informationen wie Benutzer- und Gestennamen werden per Attribute übergeben.

Eine Antwort des Servers sieht sehr ähnlich aus:

```
<GestureRecognitionAnswer>
    <LearnData user="agassi" tag="aufschlag" success="true"/>
</GestureRecognitionAnswer>
```

Hierbei wird der Typ der Anfrage wiederholt, in diesem Fall eine Klassifikationsanfrage.

6.5.1 Anfragen und Antworten

Klassifikationsanfrage

Eine Klassifikationsanfrage wird mit `ClassifyData` eingeleitet:

```
<GestureRecognitionRequest>
    <ClassifyData user="agassi">
        <Signal>
            <Value> 1.23</Value>
            <Value> 0.43</Value>
            ...
        </Signal>
        <Signal>
           ...
        </Signal>
        <Signal>
           ...
        </Signal>
    </ClassifyData>
```

```
</GestureRecognitionRequest>
```

Das Attribut `user` gibt den Benutzer an, für den die Geste erkannt werden soll. Im Anschluss werden die Werte der Signale für die Geste übermittelt.

Die Antwort darauf sieht wie folgt aus:

```
<GestureRecognitionAnswer>
    <ClassifyData user="agassi" success="true">
        <tag name="links" prob="54"/>
        <tag name="rechts" prob="46"/>
    </ClassifyData>
</GestureRecognitionAnswer>
```

Das Attribut `success` gibt an, ob die Anfrage erfolgreich bearbeitet wurde. Im Anschluss werden die zur Geste berechneten Hypothesen und ihre Wahrscheinlichkeiten übermittelt.

Lernanfrage

Eine Lernanfrage wird mit `LearnData` eingeleitet:

```
<GestureRecognitionRequest>
    <LearnData user="agassi" tag="aufschlag">
        <Signal>
            <Value> 1.23</Value>
            <Value> 0.43</Value>
            ...
        </Signal>
        <Signal>
            ...
        </Signal>
        <Signal>
            ...
        </Signal>
    </LearnData>
</GestureRecognitionRequest>
```

Das Attribut `user` gibt den Benutzer an, `tag` die Klasse, für die die Geste erlernt werden soll. Die Antwort übermittelt, ob das Lernen erfolgreich war und sieht wie folgt aus:

```
<GestureRecognitionAnswer>
    <LearnData user="agassi" tag="aufschlag" success="true"/>
</GestureRecognitionAnswer>
```

Anfragen zur Organisation der Datenbank

Weitere Anfragen dienen zur Organisation der erlernten Gesten. Als Antwort wird immer der Name der Anfrage wiederholt und zusätzlich das Attribut `success` übergeben das angibt, ob die Anfrage erfolgreich ausgeführt wurde bzw. ob das Element, nach dessen Existenz gefragt wurde, vorhanden ist.

- `<UserExists user="becker"/>` - prüft ob erlernte Gesten für den Benutzer `user` vorhanden sind.
- `<TagExists user="becker" tag="links"/>` - prüft ob die Geste `tag` für den Benutzer `user` vorhanden ist.
- `<DeleteUser user="becker"/>` - löscht alle erlernten Gesten für den Benutzer `user`.
- `<DeleteTag user="becker" tag="links"/>` - löscht die erlernte Gesten zur Klasse `tag` von Benutzer `user`.
- `<CommitData"/>` - speichert neu erlernte Gesten in der Datenbank.

6.6 Implementierungsbeispiele

Die Implementierung der Anwendungsebene liegt in C# vor und stellt die Funktionalitäten zur Verfügung, auf Sensordaten der WiiMote zu reagieren. Zum Demonstrieren und Testen der Funktionsweise wurden drei Beispielanwendungen entwickelt, die im Folgenden vorgestellt werden.

6.6.1 GestureLearner

GestureLearner (Abbildung 6.5) ist eine Oberfläche zum Erlernen von Gesten. Die Kurve der zuletzt empfangenen Bewegung wird grafisch im Hauptfenster der Anwendung angezeigt. Auf der rechten Seite der Oberfläche werden die Gesten organisiert. Man kann Gestenklassen erzeugen und benennen und diesen per Mausklick Bewegungskurven als Prototypen zuweisen. Die so erzeugte Trainingsmenge wird einem Benutzer zugeordnet und kann per Knopfdruck an den Klassifikationsserver zum Erlernen der Gesten gesendet werden. Weiterhin besteht die Möglichkeit, die aufgezeichneten Trainingsdaten zu speichern und wieder in das Programm zu laden. Der Knopf *ClassifyGesture* schickt eine Klassifikationsanfrage mit der aktuell im Hauptfenster angezeigten Bewegung an den Klassifikationsserver. *TestClassifier* dient der Validierung des Klassifikators, wobei die rechts aufgelisteten Gesten als Testmenge fungieren.

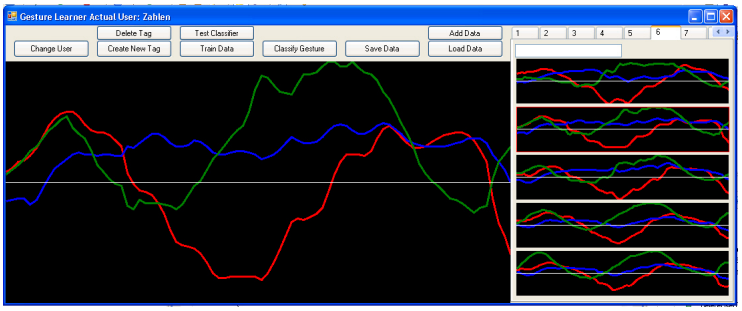


Abbildung 6.5: GestureLearner zum Aufnehmen und Trainieren von Gesten

6.6.2 Calciilator

Der Calciilator ist ein simpler Taschenrechner, der addieren und subtrahieren kann. Die Tasten des Rechners reagieren auf Gesten, die zuvor für die verschiedenen Ziffern und Operatoren erlernt wurden. Der Calciilator diente als Testanwendung der Evaluation (Abbildung 6.6).

6.6.3 Kiiboard

Diese Anwendung verbindet Gesten mit Tastatureingaben unter Windows, ähnlich dem in Abschnitt 4.1.2 vorgestellten GlovePie. Gesten werden bestimmten Tasten zugeordnet und bei Erkennen einer Geste das Drücken der zugewiesenen Taste emuliert. Anwendungen unter Windows können so mit der WiiMote als Eingabegerät gesteuert werden. Als Beispiel wurden die einzelnen Aktionen eines Tetrisspiels mit Gesten verknüpft. So konnten die herabfallenden Blöcke durch Links- bzw. Rechtsbewegungen verschoben oder durch Drehen des Handgelenks rotiert werden.

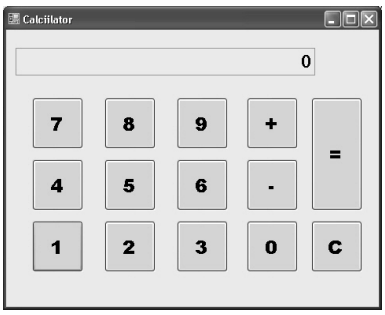


Abbildung 6.6: Calciilator. Ein mit Gesten gesteuerter Taschenrechner

6.7 Zusammenfassung

Dieses Kapitel stellte das für die Masterarbeit entwickelte Toolkit zur automatischen Klassifikation von Gesten (TaKG) vor. Das Toolkit vereinfacht die Integration von Gestensteuerung in Anwendungen und stellt die hierfür nötigen Funktionen zur Verfügung. Für die Maschinenlernfunktionalität wurde eine Serveranwendung in Java implementiert, welche die Vorverarbeitung der empfangenen Signale, Organisation der erlernten Gesten, Verwendung der Klassifikatoren und Reaktion auf Klassifikationsanfragen von Anwendungen regelt. In der Anwendungsebene werden über Schnittstellen für die gestengesteuerte Anwendung benötigte Funktionalitäten zur Verfügung gestellt. Eine davon bildet die Schnittstelle zu Sensoren und gibt mit einem Gestenobjekt die Struktur vor, in der Sensordaten beschrieben werden. Für die WiiMote und die von ihr gemessenen Beschleunigungswerte ist eine Implementierung angepasst an diese Schnittstelle vorhanden. Die andere Funktionalität regelt die Kommunikation mit dem Klassifikationsserver und ermöglicht so aus Anwendungsprogrammen heraus ohne großen Aufwand Klassifikationsanfragen zu stellen.

Kapitel 7

Evaluation

7.1 Ziel der Evaluation

Für die Evaluation wurde mit einer Reihe von Probanden Versuche durchgeführt mit dem Ziel, Testdatensätze zu sammeln. Auch wenn die Evaluation mit einer Anzahl von 15 Probanden nicht signifikant sein kann, so zeichnen sich erste Richtungen und Ergebnisse ab, in die eine ausführlichere Evaluation führen kann. Um die Ergebnisse trotzdem möglichst repräsentativ zu halten, wurden Alter und Geschlecht der Probanden balanciert. Hierdurch sollte verhindert werden, dass nicht nur jüngere Menschen, die evtl. sogar schon Erfahrungen mit der Wii Spielekonsole gemacht haben, an dem Test teilnehmen würden. Insbesondere in Hinblick auf das i2home Projekt, bei dem Bedienkonzepte für ältere Personen mit Handicaps entwickelt werden, sollten auch Senioren in die Tests miteinbezogen werden, um so etwaige Probleme zu finden, die bei der Bedienung durch jüngere Personen nicht auftreten.

Folgende Probanden nahmen (in Altersgruppen unterteilt) an dem Versuch teil:

1. 5 Studenten im Alter von 20-30 Jahren
2. 3 Mitarbeiter des DFKI im Alter von 25-50 Jahren
3. 3 Probanden zwischen 50-80 Jahren
4. 4 Senioren ab 80 Jahre (davon 3 älter als 90 Jahre)

An die Personen ab 50 sind wir im Rahmen eines Wii-Workshops herangetreten. Dieser fand zum Teil am DFKI statt, zum Teil im Altenwohnstift Egon-Reinert-Haus in Saarbrücken. Im Rahmen des Workshops, in dem die Forschungsziele

des i2home-Projekts vorgestellt wurden, haben sich mehrere Personen bereiterklärt, an dem Versuch teilzunehmen.

Mit der Evaluation der in dieser Arbeit entwickelten Gestenklassifikation wurden mehrere Ziele verfolgt. Besonders von Interesse war zunächst zu testen, mit welcher Präzision die Erkennungsalgorithmen arbeiten und Vergleiche zwischen den unterschiedlichen Merkmalsextraktionsverfahren und Maschinenlernverfahren zu ziehen. Die aufgezeichneten Testdaten wurden daher mit allen Kombinationen aus Extraktionsverfahren und Lernverfahren ausgewertet um zu sehen, welche der Kombinationen die höchsten Erkennungsraten erzielt. Weiterhin sollte beobachtet werden, wie gut die Probanden unterschiedlichster Alterstufen mit dem Prinzip der Maschinensteuerung mit Gesten umzugehen verstanden.

Das Sammeln der Daten von mehreren Personen bietet weiterhin die Möglichkeit, die Klassifikatoren daraufhin zu untersuchen, inwiefern die Gesten für eine Wiedererkennung generalisiert werden können. Dafür werden alle für jeden Probanden individuell aufgezeichneten Trainingsdaten zusammengefasst und untersucht, wie gut Gesten anhand dieser allgemeinen Trainingsmenge erkannt werden. Vorstellbar ist, dass bei guten Ergebnissen und ausreichend großen Wissensbasen die Trainingsphase später wegfallen kann.

Während der Evaluationsphase hat sich herausgestellt, dass es schwierig ist, im Rahmen einer Masterarbeit Tests durchzuführen, die alle für die Untersuchung interessanten Kriterien abdecken. Neben den Merkmalsextraktionsverfahren, deren Güte einen wichtigen Teil zur Präzision der Erkennung beiträgt, spielen eine Reihe von anderen Faktoren eine entscheidende Rolle. Um weitere aufschlussreiche Ergebnisse zu erhalten, müssten zusätzlich in den psychologischen Bereichen der Kognition und Motorik Beobachtungen durchgeführt und das Eingabegerät ergonomisch untersucht werden.

7.2 Versuchsablauf

7.2.1 Szenario

Während des Versuchs soll ein Taschenrechner mit Hilfe von mit der WiiMote aufgezeichneten Gesten gesteuert werden. Der Taschenrechner enthält Tasten für die Ziffern 0-9, die Operatoren Plus, Minus und Gleich und eine Clear-Taste. Durch in die Luft Zeichnen der zugehörigen Ziffern und Operatoren, vergleichbar mit dem Schreiben an eine Tafel, werden die verschiedenen Tasten des Rechners bedient. Hierbei ist die Form der Schriftzüge nicht vorgegeben, es gibt lediglich die Bedingung, dass die Bewegungen zusammenhängend sein müssen. Die Gesten werden für jeden Benutzer individuell erlernt und anschließend bei

der Erkennung den entsprechenden Tasten zugeordnet. Der Gestenwortschatz hat eine Größe von 13 (1, 2, 3, 4, 5, 6, 7, 8, 9, 0, +, -, =). Die Clear-Taste wird bei dem Test nicht verwendet.

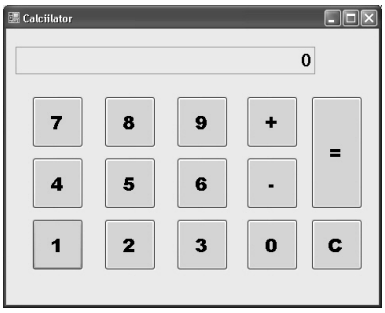


Abbildung 7.1: Taschenrechner-Anwendung der Evaluation

7.2.2 Ablauf

Zunächst wurde den Probanden das Konzept der Maschinensteuerung mit Gesten vorgestellt und demonstriert, wie der Taschenrechner bedient werden kann. Anschließend erhalten sie die Möglichkeit, sich mit der Bewegungsaufzeichnung über die WiiMote vertraut zu machen. Hierbei bedient der Proband die WiiMote im Sitzen ohne den Arm abzustützen. Für die Trainingsphase wird die in Kapitel 6.6 vorgestellte Oberfläche *GestureLearner* verwendet. Während deren Verlauf wird zu jeder Geste die zugehörige Bewegung dreimal aufgezeichnet. Nachdem die Gesten erlernt wurden, hat der Proband kurz die Möglichkeit, die Bedienung des Taschenrechners zu testen. Dann erfolgt die Aufzeichnung der Daten. Der Proband wird darum gebeten, drei Gleichungen in den Taschenrechner einzugeben, die aus insgesamt 42 Zeichen bestehen. Die Gleichungen umfassen jede Geste mindestens dreimal und lauten wie folgt:

$$\begin{aligned} 56 + 743 - 1029 &= \\ 111 - 249 + 357 + 608 &= \\ 9163 - 204 + 587 &= \end{aligned} \tag{7.1}$$

Da mit der Evaluation die Präzision der Klassifikation getestet werden soll, werden während der Aufzeichnung falsch erkannte Gesten nicht wiederholt. Die Eingaben werden aufgezeichnet und gespeichert womit die Möglichkeit gegeben wird, anschliessend anhand der Testdaten die verschiedenen Kombinationen aus Merkmalsextraktionsverfahren und Maschinenlernverfahren zu testen.

7.3 Auswertung

7.3.1 Messergebnisse

Tabelle 7.3.1 listet prozentual die korrekt erkannten Gesten auf. Hierbei sind in den Spalten die verschiedenen Kombination aus Maschinenlern- und Merkmalsextrakionsverfahren aufgelistet. Als Klassifikatoren wurden die in Abschnitt 3.3.2 vorgestellte Support Vector Machine (SVM) und ein Multilayer-Perceptron-Netz (NN), als Merkmalsextraktionsverfahren die in Kapitel 5 aufgelisteten Verfahren Äquidistantes Gitter und Attributebeschreibendes Verfahren und zusätzlich ein Äquidistantes Gittermodell verwendet, das mit auf das Intervall [0,1] normalisierten Werten arbeitet.

An den Ergebnissen lässt sich ablesen, dass die Erkennungsrate mit zunehmendem Alter abnimmt, woran man einen Zusammenhang zwischen der Erfahrung im Umgang mit modernen Eingabemethoden und der Richtigkeit der Ergebnisse erkennen kann. Sicherlich ist die Lernkurve für den Umgang mit der WiiMote bei jüngeren Personen steiler, wodurch ihre besseren Ergebnisse zusätzlich erklärt werden können.

Bei den Senioren sind die Ergebnisse insgesamt am schlechtesten ausgefallen und liegen größtenteils um die 50 Prozent. Dies liegt zum Teil daran, dass sie aufgrund ihrer körperlichen Einschränkungen die Gesten nicht immer sorgfältig ausführen konnten. So haben sich besonders zum Ende des Experiments die Ergebnisse verschlechtert, da die Konzentration der Probandinnen nachließ oder ihre Arme schwer wurden, so dass ihre Hände anfingen zu zittern.

7.3.2 Vergleich der Algorithmen

Während der Auswertung fand sich keine Kombination aus Extraktions- und Maschinenlernverfahren, die durchgehend die besten Ergebnisse erzielt hat. So hat selbst das Verfahren mit dem Attributbeschreibendem Modell, das insgesamt am schlechtesten abgeschnitten hat, für einzelne Benutzer (m2, m4) die größte Genauigkeit erreicht.

Vergleicht man die Maschinenlernverfahren direkt miteinander, so erzielt die Support Vector Machine meistens die besseren Erkennungsresultate. Dies und die auffällig höhere Geschwindigkeit der SVM machen sie zu der besseren Wahl beider getesteter Verfahren. Bei den Merkmalsextraktionsverfahren waren die Resultate der beiden Gittermodelle besser als die des Attributbeschreibenden Modells, wobei die nicht normalisierten Werte insgesamt zu genaueren Ergebnissen führten. Lediglich bei den Senioren führte eine Normalisierung der Mess-

Tabelle 7.1: Ergebnisse des Vergleichs zwischen den verschiedenen Kombinationen aus ML-Algorithmen und Merkmalsextraktionsverfahren. Die besten Ergebnisse sind jeweils fett gedruckt.

User	Gitter		Attribute		norm. Gitter	
	SVM	NN	SVM	NN	SVM	NN
Studenten						
m1	76%	81%	79%	79%	81%	**86%**
m2	**100%**	93%	**100%**	**100%**	**100%**	98%
w1	**93%**	**93%**	88%	86%	81%	90%
w2	**76%**	64%	55%	55%	71%	64%
m3	88%	**90%**	74%	69%	81%	83%
Durchschnitt	**87%**	84%	79%	78%	83%	84%
Standardabweichung	10%	12%	17%	17%	10%	12%
DFKI-Mitarbeiter						
m4	**79%**	69%	**79%**	67%	64%	57%
m5	**74%**	62%	62%	60%	69%	64%
w3	83%	83%	71%	60%	**88%**	86%
Durchschnitt	**79%**	71%	71%	62%	74%	69%
Standardabweichung	3%	5%	8%	4%	5%	6%
Probanden 50-80 Jahre						
w4	**83%**	81%	60%	67%	76%	79%
w5	60%	57%	52%	48%	**62%**	52%
w6	48%	**55%**	40%	36%	52%	50%
Durchschnitt	**64%**	**64%**	51%	50%	63%	60%
Standardabweichung	18%	15%	10%	16%	12%	16%
Senioren						
w7	71%	67%	52%	62%	**76%**	**76%**
w8	45%	38%	45%	43%	**50%**	40%
w9	**55%**	**55%**	62%	60%	48%	50%
w10	57%	43%	50%	40%	**76%**	64%
Durchschnitt	57%	51%	52%	51%	**63%**	58%
Standardabweichung	11%	13%	7%	11%	16%	16%
Gesamtschnitt	**73%**	69%	65%	62%	72%	69%
Standardabweichung	16%	18%	17%	17%	15%	17%

werte zu einer höheren Präzision. Dies kann daran liegen, dass die Senioren ihre Gesten möglicherweise mit unterschiedlicher Intensität ausgeführt haben (bei gleich bleibendem Bewegungspfad), so dass eine Normalisierung der Werte einen Vorteil bringt.

7.3.3 Test einer allgemeinen Wissensbasis

Ein weiteres Ziel der Evaluation war herauszufinden, inwiefern man anhand aller Trainingsdatensätze eine allgemeine Wissensbasis erstellen kann. Sind die Ergebnisse gut und hat man eine genügend große Anzahl an Trainingsdaten, könnte in Zukunft die Trainingsphase ausgelassen werden. Für den Versuch wurden die erlernten Zahlengesten aller Probanden zusammen als Trainingsdatensatz verwendet und mit Hilfe der Gesten der aufgenommen Gleichungen evaluiert. Tabelle 7.2 vergleicht die Erkennungsrate bei der Verwendung von allgemeinen und speziellen Gesten.

Tabelle 7.2: Vergleich der Anzahl korrekt erkannter Gesten zwischen allgemeiner und spezieller Trainingsmenge

User	allgemeine Trainingsmenge		spezialisierte Trainingsmenge
	Gitter	Normalisiertes Gitter	Gitter
m1	83%	**90%**	76%
m2	85%	88%	**98%**
w1	85%	88%	**93%**
w2	47%	52%	**76%**
m3	78%	**88%**	**88%**
m4	26%	33%	**79%**
m5	64%	62%	**74%**
w3	45%	50%	**83%**
w4	45%	38%	**76%**
w5	45%	48%	**60%**
w6	47%	**48%**	**48%**
w7	33%	48%	**71%**
w8	50%	**55%**	45%
w9	35%	45%	**55%**
w10	42%	52%	**57%**
Mittelwert	54%	59%	**72%**

Die Tests wurden mit der Support Vector Machine ausgeführt und die korrekt erkannten Ergebnisse zwischen Gitter und normalisiertem Gitter miteinander verglichen. Im Schnitt lagen die korrekt erkannten Gesten beim unnormalisierten Gitter bei 54% und beim normalisierten bei 59%. Die einzelnen Ergebnisse

beim normalisierten Gitter sind beinahe durchgehend besser. Erklären lässt sich dieser Umstand dadurch, dass die Ziffern von den einzelnen Probanden meist ähnlich geschrieben werden, jedoch die Intensität der Bewegung und damit die Amplitudenwerte der Beschleunigungssensoren variieren. Eine Normalisierung der Werte schwächt diese Unterschiede ab.

Die spezialisierten Gesten liefern eindeutig die präziseren Erkennungsresultate. Dies war zu erwarten, da die Bewegungen zu den Gesten nicht vorgegeben waren und die Trainingsdaten bei allgemeiner Trainingsmenge für jede Geste verschiedene Protoypen enthalten. Hierdurch wird die Gefahr der Verwechslung vergrößert.

7.3.4 Hypothesengitter

Weka ermöglicht es, die Hypothesenreihenfolge sortiert nach ihrer Wahrscheinlichkeit zu ermitteln. In Abbildung 7.2 ist aufgelistet, wie viele der korrekten Gesten an welcher Stelle der Hypothesenrangfolge auftauchen. Hierbei sind wie erwartet die meisten Ergebnisse auf Rang eins. Einige Ergebnisse tauchen erst später in der Rangfolge auf. Dies sind vor allem die Gesten, die falsch ausgeführt wurden. Ein Großteil der korrekten Ergebnisse siedeln sich jedoch im Bereich von Rang zwei bis fünf an. Es wurden also Gesten gefunden, die der Klassifikator ähnlicher als die korrekte Geste eingeordnet hat. Die Suche nach den Ursachen hierfür lässt vermuten, dass sich einige Gesten so sehr ähneln, dass sie miteinander verwechselt werden können. Im nächsten Abschnitt wird daher untersucht, welche Gesten besonders häufig miteinander verwechselt wurden.

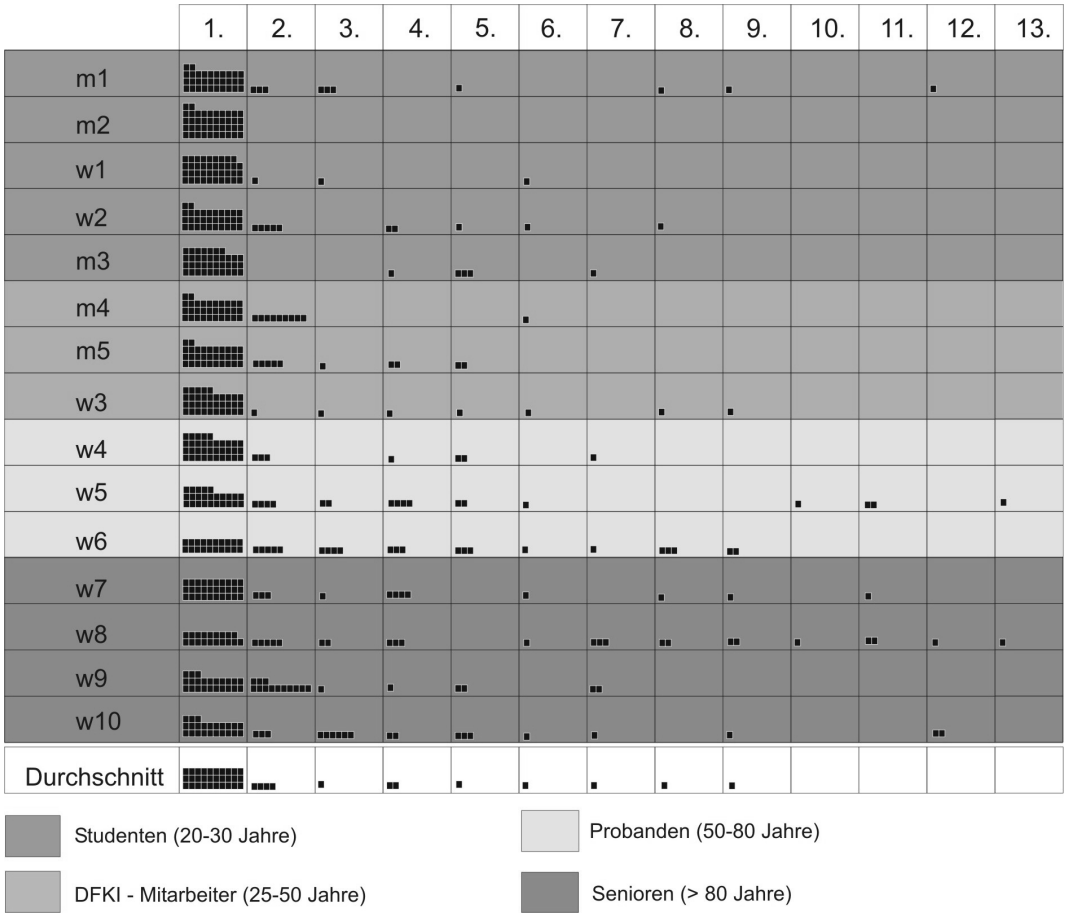


Abbildung 7.2: Anzahl der an n-tem Rang der Hypothesenrangfolge korrekt klassifizierten Gesten. Am häufigsten wurden die Ziffern direkt an erster Stelle erkannt. Falsch erkannte Gesten sind größtenteils unter den ersten fünf Kandidaten zu finden. Ausreißer sind vor allem dadurch zu erklären, dass Gesten anders ausgeführt als sie trainiert wurden.

7.3.5 Übersicht über häufig verwechselte Gesten

Im Folgenden wird untersucht, welche Gesten am häufigsten miteinander verwechselt wurden. Hierzu wurden die Daten der Versuchspersonen m1, m2, m3, m4, m5, w1, w2 und w9 verwendet, da hier die meisten falsch klassifizierten Gesten im vorderen Bereich der Hypothesenrangfolge auftauchen. In Abbildung 7.3 wird die Häufigkeit der unterschiedlichen Verwechslungen aufgelistet.

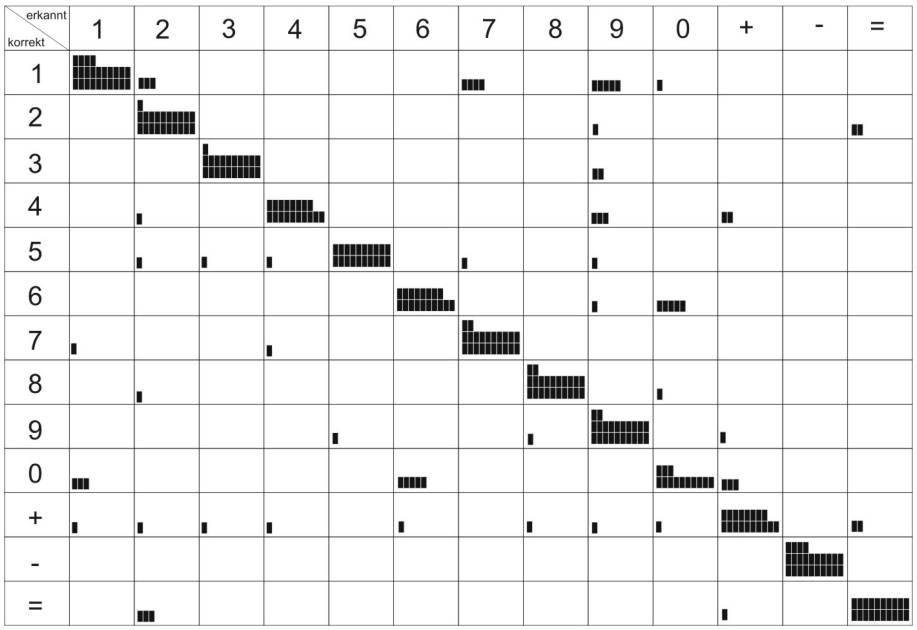


Abbildung 7.3: Die Tabelle listet auf, wie häufig die Gesten eines Typs den unterschiedlichen Ziffern zugeordnet wurden.

Tabelle 7.3: Anzahl der Verwechslungen der am häufigsten miteinander vertauschten Zahlenpaare

Verwechslungspaar	Anzahl Verwechslungen
0↔6	10
1↔7	6
2↔=	5
1↔9	5
0↔+	4
1↔2	3
4↔9	3

In Tabelle 7.3 sind die Gestenpaare aufgelistet, die am häufigsten miteinander verwechselt wurden. Die dazu am häufigsten ausgeführten Schriftzüge sind in Abbildung 7.4 dargestellt. Hierdurch wird ersichtlich, weshalb die Verwechslungen zustande kamen. Besonders bei den Gesten (0,6), (1,7), (2,=) und (4,9)

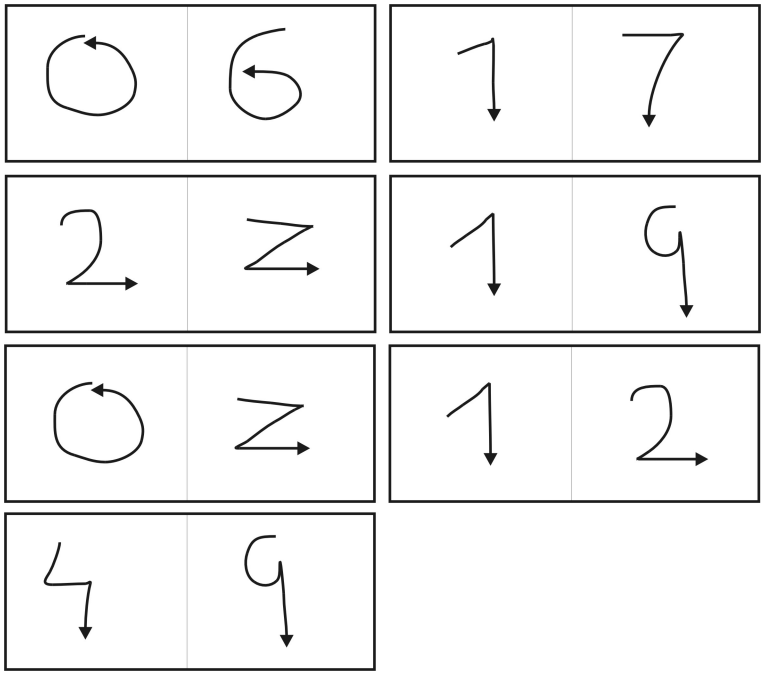


Abbildung 7.4: Häufig miteinander verwechselte Gesten. Wie man sieht, sind die Bewegungsabläufe bei vielen Ziffern, die häufiger miteinander verwechselt wurden, ähnlich. Betrachtet man zum Beispiel die Bewegung zu den Ziffern 0 und 6, so ist der einzige Unterschied eine länger andauernde Aufwärtsbewegung für die Ziffer 0. 1 und 7 unterscheiden sich geringfügig am Anfang der Bewegung.

sind sich die Bewegungen so ähnlich, dass kleine Abweichungen von der erlernten Geste falsch gedeutet werden können. Betrachtet man zum Beispiel die 0, so ist der einzige Unterschied zur 6, dass die Aufwärtsbewegung zum Schluss der Geste ein wenig kürzer ist.

Die Verwechslung von sich zu sehr ähnelnden Gesten scheint ein Hauptgrund für Falschklassifikationen zu sein. Daher sollten im Vorhinein bei der Auswahl der Gesten sich möglichst gut voneinander unterscheidende Bewegungen gewählt werden. Ein ähnliches Problem tritt bei der Handschrifterkennung auf. Beim PDA von Palm wurde dies gelöst, indem die Gesten für die einzelnen Eingaben vorgegeben werden und zwar so, dass zu große Ähnlichkeiten möglichst vermieden werden (siehe Anhang A).

7.4 Äußere Einflüsse auf die Erkennungsresultate

Falschklassifikationen sind nicht ausschließlich mit Fehlern bzw. Verwechslungen der Klassifikatoren zu begründen. Während der Versuche konnten eine Reihe äußerer Einflüsse auf die Genauigkeit der Gestenerkennung beobachtet werden.

Generell hat eine längere Übungsphase die Erkennungsresultate deutlich verbessert. Leider war dies, insbesondere im Altenstift, aufgrund eines engen Zeitplans nicht immer möglich. Ein weiterer Grund war, dass die Gesten mit Dauer der Versuche weniger sorgfältig ausgeführt wurden. Grund hierfür war einerseits die beobachtbar abnehmende Konzentration der Probanden, andererseits traten, besonders nach Aussage der Senioren, Ermüdungserscheinungen in den Armen auf. Zusätzlich kam es vor, dass Probanden sich nicht mehr genau an die erlernten Prototypen zu den Gesten erinnern konnten, so dass z.B. die Kreisbewegung zu einer null gegen anstatt im Uhrzeigersinn ausgeführt wurde.

Ein gelegentlich auftretendes Problem war, dass sich die Grundhaltung der WiiMote zwischen Trainingsphase und Evaluationsphase unterschied. Dadurch dass die WiiMote während der Aufnahme normal nach vorne gerichtet wurde, während der Anwendung anschließend jedoch seitlich, wurden verschiedene Sensoren belastet, was eine korrekte Erkennung der Geste nich mehr möglich machte.

7.5 Zusammenfassung

Dieses Kapitel befasset sich mit der Evaluation der implementierten Gestenklassifikation. Für die Evaluation wurden Versuche mit Personen unterschiedlichster Altersklassen durchgeführt und wegen des i2home-Projekts der Schwerpunkt besonders auf ältere Personen gelegt. Während der Tests wurden Gesten zu Ziffern und Operatoren erlernt und die Präzision der Gestenklassifikation ermittelt. Folgende Aspekte wurden untersucht:

- **Anwendbarkeit der Maschinensteuerung mit Gesten**
  Nach kurzer Eingewöhnungszeit gelang es Probanden aller Alterstufen, mit der WiiMote umzugehen. Selbst über 90 Jahre alte Probanden waren somit in der Lage, die Testanwendung über Gesten zu steuern.

- **Vergleich der Algorithmen**
  Es wurden die verwendeten Support Vector Machine und Neuronale Netze

Maschinenlernverfahren sowie die unterschiedlichen Merkmalsextraktionsverfahren miteinander verglichen. Hierbei hat sich herausgestellt, dass im Schnitt der SVM-Algorithmus kombiniert mit dem äquidistanten Gittermodell (siehe Kapitel 5.2.2) die besten Resultate lieferte. Mit 73% korrekt erkannten Gesten und einer Standardabweichung von 16% ist dieses Ergebnis vergleichbar mit den Resultaten des Dynamic Time Warp Algorithmus gewesen, der beim XWand-Projekt [Wilsona and Wilson, 2004] in der Testphase eine Genauigkeit von 71% erzielte. Das beste Ergebnis dieser Kombination lag bei 100%, das schlechteste bei 45%.

- **Verwendung einer allgemeinen Trainingsmenge**
  Hierbei wurde untersucht, wie gut die Erkennungsresultate bei der Zusammenlegung aller Trainingsdaten zu einer allgemeinen Trainingsmenge wurden. Die korrekt erkannten Resultate lagen im besten Fall bei 59%. Im Vergleich hierzu erlangte man bei speziellen Trainingsmengen im Schnitt 72%.

- **Untersuchung der Hypothesenrangfolge**
  Ein Blick auf die Hypothesenrangfolge ergab, dass bei vielen falsch erkannten Gesten das korrekte Ergebnis recht früh in der Hypothesenrangfolge auftauchten. Das ließ die Vermutung zu, dass sich einige Gesten sehr ähnlich waren und deswegen miteinander verwechselt wurden.

- **Untersuchung der verwechselten Gesten**
  Es wurde gezählt, wie häufig verschieden Gestenpaare miteinander verwechselt wurden. In den häufigsten Fällen waren dies Gesten von Ziffern und Operatoren, deren Bewegungspfade sich ziemlich ähnlich sind. Beispielsweise unterscheiden sich die Ziffern 0 und 6 lediglich durch eine unterschiedlich lange Aufwärtsbewegung am Ende der Geste. Ähnlichkeiten wurden u.a. auch zwischen den Ziffern (1,7) und (4,9) gefunden. Die zu große Ähnlichkeit von unterschiedlichen Gesten war ein Hauptgrund für Falschklassifikationen und könnte umgangen werden, indem man sich genügend voneinander unterscheidende Gesten verwendet.

Kapitel 8

Zusammenfassung und Ausblick

Das i2home Projekt des Instituts für Intelligente Benutzerschnittstellen des DFKIs verwendet eine Vielzahl an Eingabemöglichkeiten, um dem Menschen die Kommunikation mit dem Computer zu vereinfachen. Die Benutzer werden hierbei durch eine über Touchscreen und Spracheingabe steuerbare übersichtliche Menüführung bei der Bedienung unterstützt. Ein Ziel ist, Menschen mit kognitiven Behinderungen oder älteren Menschen die Verwendung moderner Technologien zu ermöglichen. Hierbei kann es aufgrund ihrer Einschränkungen zu Schwierigkeiten bei der Bedienung kommen, weshalb von Vorteil ist, eine große Anzahl an Bedienkonzepten zur Verfügung zu haben.

In dieser Arbeit wurde ein Toolkit entwickelt, das die multimodale Kommunikation zwischen Computer und Benutzer um den Aspekt der Steuerung mit Gesten erweitert. Als Eingabegerät wurde die WiiMote von Nintendo verwendet, in den Beschleunigungssensoren für jede Dimension des Raums eingebaut sind. Es wurden Algorithmen entwickelt, um die Signale der WiiMote auszulesen und auszuwerten, damit sie als Gesten maschinell erlernt und wiedererkannt werden können. Weiterhin wurden Beispielapplikationen realisiert, die mit Hilfe der erlernten Gesten gesteuert werden können. Zusätzlich wurde das Toolkit TaKG entwickelt, das die Erstellung gestengesteuerter Anwendungen ermöglicht.

8.1 Bearbeitete Fragen

- **Verwendung des Wii-Controllers als Eingabegerät**
  Es gibt mehrere frei verfügbare Bibliotheken, die den Zugriff auf den Wii-Controller bereits implementieren. In der Arbeit wird das für C# geschriebene WiimotLib verwendet. Grund der Entscheidung für diese Bibliothek ist, dass alle zur Zeit bekannten Funktionalitäten der WiiMote (ausser der

Tonausgabe) unterstützt werden. Weiterhin ist die Bibliothek gut dokumentiert.

- **Merkmalsextraktion**
  Für die Merkmalsextraktion aus den Beschleunigungssignalen wurden zwei verschiedene Ansätze entwickelt. Der erste Ansatz ist das Äquidistante Gitter und beschreibt die Kurven über eine feste Anzahl an Abtastpunkten, deren Werte durch lineare Interpolation der Umgebung ermittelt werden. In einer abgeänderten Variante werden hierbei die Werte zuvor normalisiert. Im zweiten Ansatz, dem Attributbeschreibendem Modell, werden die Kurven durch mathematische Eigenschaften wie Mittelwert, Extremstellen oder Standardabweichung beschrieben.

- **Erlernen und Wiedererkennen von Gesten**
  Für das Erlernen und Wiedererkennen der Gesten wurde die Maschinenlern-Bibliothek Weka verwendet. Hierfür musste die Funktionsweise von Weka erarbeitet und die Daten so beschrieben werden, dass Weka sie verarbeiten kann. Als Klassifikationsverfahren wurden ein *Multilayer-Perceptron-Netz* und eine *Support Vector Machine* mit dem *Sequential Minimal Optimization*-Algorithmus verwendet.

- **Erstellen eines Toolkits zur Klassifikation**
  Es wurde das Toolkit TaKG entwickelt, mit dessen Hilfe der Einbau einer Gestensteuerung in Anwendungen vereinfacht werden kann. Es besteht zum einen aus der Klassifikationsebene, die über eine Serveranwendung verschiedene Funktionen wie das Erlernen und das Klassifizieren einer Geste zur Verfügung stellt. Zum anderen werden auf der Anwendungsebene Schnittstellen zur Verfügung gestellt, die den Zugriff auf die WiiMote vereinfachen und Methoden, die die Kommunikation mit der Klassifikationsebene organisieren. Das Toolkit ist konzeptionell so aufgebaut, dass über vordefinierte Schnittstellen einzelne Komponenten ausgetauscht werden können und somit z.B. die Möglichkeit besteht, einen anderen Sensor als die WiiMote als Eingabegerät zu verwenden. Weiterhin lässt sich der Klassifikationsalgorithmus austauschen, sofern sich an die Schnittstellenvorgaben gehalten wird.

- **Evaluation**
  Für die Evaluation der Gestenerkennung wurde ein Wii-Workshops veranstaltet. Im Rahmen hiervon wurde Senioren des Altenwohnstift Egon-Reinert-Haus in Saarbrücken und weiteren Probanden aller Altersstufen die Idee gestengesteuerter Anwendungen vorgestellt und mit ihnen anschließend über eine Beispielanwendungen Testdaten gesammelt.

  Ein Ziel der Evaluation war, die Präzision der unterschiedlichen Kombinationen aus Maschinenlernverfahren und Merkmalsextraktionsverfahren

miteinander zu vergleichen. Dabei stellte sich heraus, dass die verwendete Support Vector Machine um 3% mehr Gesten richtig erkannt hat, als das Multilayer-Perceptron-Netz. Von den Merkmalsextraktionsverfahren erkannte das Modell mit Äquidistanten Gittern kombiniert mit der SVM die meisten Gesten richtig, die Präzision lag bei 72%. Eine Untersuchung der miteinander verwechselten Gesten ergab, dass besonders sich stark ähnelnde Bewegungen häufig miteinander verwechselt wurden. Bei solchen Gesten war jedoch ein frühes auftauchen der korrekten Geste in der Hypothesenrangfolge zu beobachten.

Es sollte weiterhin beobachtet werden, wie gut die Probanden mit der Gestensteuerung umgehen können. Das Ergebnis war überraschend gut, selbst die Senioren konnten nach kurzer Eingewöhnungszeit die WiiMote bedienen. Lediglich mit Dauer des Experimentes machten sich bei ihnen körperliche Einschränkungen bemerkbar, zum Beispiel durch Zittern der Hand oder durch Verlieren der Konzentration. Ein zusätzliches Problem war, dass sich die Probanden während der Anwendung nicht mehr an die trainierten Gesten erinnern konnten. So wurden die Gesten während der Anwendungsphase unterschiedlich ausgeführt als in der Lernphase, was zu einigen Fehlern in den Erkennungsresultaten führte.

8.2 Ausblick

Die vorliegende Arbeit liefert einen ersten Ansatz der Gestenerkennung. Die Gesten werden einzeln erkannt und per Knopfdruck (*push-to-move*) die Ausführung einer Geste signalisiert. Vergleicht man den Entwicklungsstatus mit der Spracherkennung, so wäre man an einem Punkt, an dem einzelne per *Push-To-Talk* aufgezeichnete Wörter erkannt würden. Selbstverständlich sind moderne Spracherkennungssystem um einiges ausgereifter. Einige Lösungen der hierbei auftretenden Probleme geben Anregungen um die Erkennung von Gesten weiterzuentwickeln [Berton et al., 2006]. Am *Georgia Institute of Technology* wurde z.B. ein Spracherkennungs-Toolkit der Universität in Cambridge verwendet, um bestimmte Klassen von Gesten zu erkennen [Westeyn et al., 2003].

8.2.1 Test weiterer Mustererkennungsverfahren

Aufgrund der einfachen Implementierungsmöglichkeit von Neuronalen Netzen und Support Vector Machines war Weka die Wahl für die ersten Gestenerkennungsversuche und hat bereits gute Ergebnisse erzielt. In der Spracherkennung haben sich Verfahren wie die in Abschnitt 5.1 vorgestellten *Dynamic Time Warp*

und vor allem *Hidden Markov Models* als praktikabel erwiesen. Es bleibt zu untersuchen, ob eine Verwendung dieser Verfahren im TaGK-Toolkit die Erkennungsrate verbessern könnte.

8.2.2 Segmentierung von Gesten

Moderne Spracherkennungsysteme erkennen automatisch Anfang und Ende von Sprache, wodurch die *push-to-talk*-Methode abgelöst wurde. Die Aufgabe erweist sich insofern als schwierig, da ständig Hintergrundgeräusche die Signale beeinflussen und der Computer zwischen Hintergrundgeräuschen und Spracheingabe unterscheiden muss. Für die Realisierung wird der Geräuschpegel untersucht und nur Aufnahmen über einem Lautstärkeschwellenwert verarbeitet.

In dieser Arbeit wird die Ausführung einer Geste dadurch signalisiert, dass während der Bewegung ein Knopf gedrückt wird, also durch ein *push-to-move* Verfahren vergleichbar mit dem *push-to-talk* Verfahren bei der Spracherkennung. Durch Zittern der Hand oder andere ungewollte Bewegungen entsteht ein Rauschen, das die automatische Erkennung von Anfang und Ende einer Gesteneingabe erschwert. Hier kann mit Hilfe eines Schwellenwertes für die Beschleunigungswerte festgelegt werden, wann eine Bewegung als Eingabegeste gewertet werden soll. Es entsteht das zusätzliche Problem, dass zufällig ausgeführte Bewegungen ignoriert werden müssen.

8.2.3 Diskrete und kontinuierliche Gesten

Frühe Spracherkennungssystem waren nicht in der Lage, Sätze zu erkennen ohne dass wahrnehmbare Pausen zwischen den einzelnen Worten eingebaut wurden. Der Mensch spricht jedoch kontinuierlich ohne Pausen, so dass die Worte ineinander übergehen, wodurch das zusätzliche Problem auftritt, dass Worte in Sätzen anders ausgesprochen werden, als würden sie einzeln verwendet werden.

Ähnliche Schwierigkeiten ergeben sich, wenn man kontinuierliche Bewegungen erkennen will. Ein Szenario hierfür wäre zum Beispiel das Erkennen von Zeichensprache [Starner et al., 1998]. Hier sollen die einzelnen Gesten erkannt werden, ohne dass man künstliche Pausen in die Bewegung einbaut. Zusätzlich muss berücksichtigt werden, dass aufgrund der Übergänge die Gesten auf andere Art und Weise ausgeführt werden, als würde man sie einzeln verwenden.

8.2.4 Festlegen des Gestenwortschatzes

In der Evaluation stellte sich heraus, dass sich zu sehr ähnelnde Gesten häufig miteinander verwechselt werden. Gerade bei der Erzeugung künstlicher Gesten erhöht daher die Verwendung von sich unterscheidenden Bewegungen die Präzision der Erkennungsresultate. Die Buchstabenerkennung des PDA von Palm (Anhang A) ist ein gutes Beispiel für das Vordefinieren eines Gestenwortschatzes. Hier sind die Strichzüge zu den einzelnen Eingabezeichen vorgegeben, die klar voneinander unterschieden werden können und zusätzlich recht einfach gehalten werden. Für die Erzeugung eines künstlichen Alphabets wäre ein Maß hilfreich, das die Ähnlichkeit zwischen Bewegungen misst und mit dessen Hilfe sich zu ähnliche Gesten verworfen werden können. Aufgrund der vorgegebenen Bewegung können die Trainingsdaten der Gesten zusätzlich konventionalisiert werden, ein spezielles Training für einzelne Benutzer würde wegfallen.

Bei der Gestenerkennung im SmartKom-Projekt entfernt man sich von der Idee der künstlichen Gesten. Hier wird vorgegeben, dass die zu verwendenden Gesten natürlich sein sollen, um dem Benutzer eine intuitive Bedienung zu ermöglichen, ohne zuvor mit ihm Gesten erlernen zu müssen [Racky et al., 2006]. Bei einer Beschränkung auf einen natürlichen Gestenworschatz können so vom Menschen während der Kommunikation unbewusst ausgeführte Gesten gedeutet werden [Stern et al., 2006].

8.2.5 Kontextbezogene Hypothesenanalyse

Der Klassifikator gibt nicht nur die beste Hypothese für eine Geste zurück, sondern eine Auflistung seiner Hypothesen sortiert nach ihrer Ähnlichkeit mit den Prototypen. Diese Information kann man nutzen, um im Zusammenhang mit dem aktuellen Anwendungskontext herauszufinden, welche der vermuteten Gesten am ehesten ausgeführt wurde. So kann es vorkommen, dass die beste Hypothese im aktuellen Anwendungskontext keinen Sinn ergibt, die Hypothese an zweiter Stelle jedoch schon. Zur Analyse der Hypothesen kann also eine Art Bewertungsfunktion erstellt werden, die anhand des aktuellen Kontexts und dem Ergebnis des Klassifikators eine genauere Vermutung über die Eingabegeste trifft, als wenn ausschließlich das Ergebnis der Gestenerkennung verwendet würde [Engel and Pfleger, 2006].

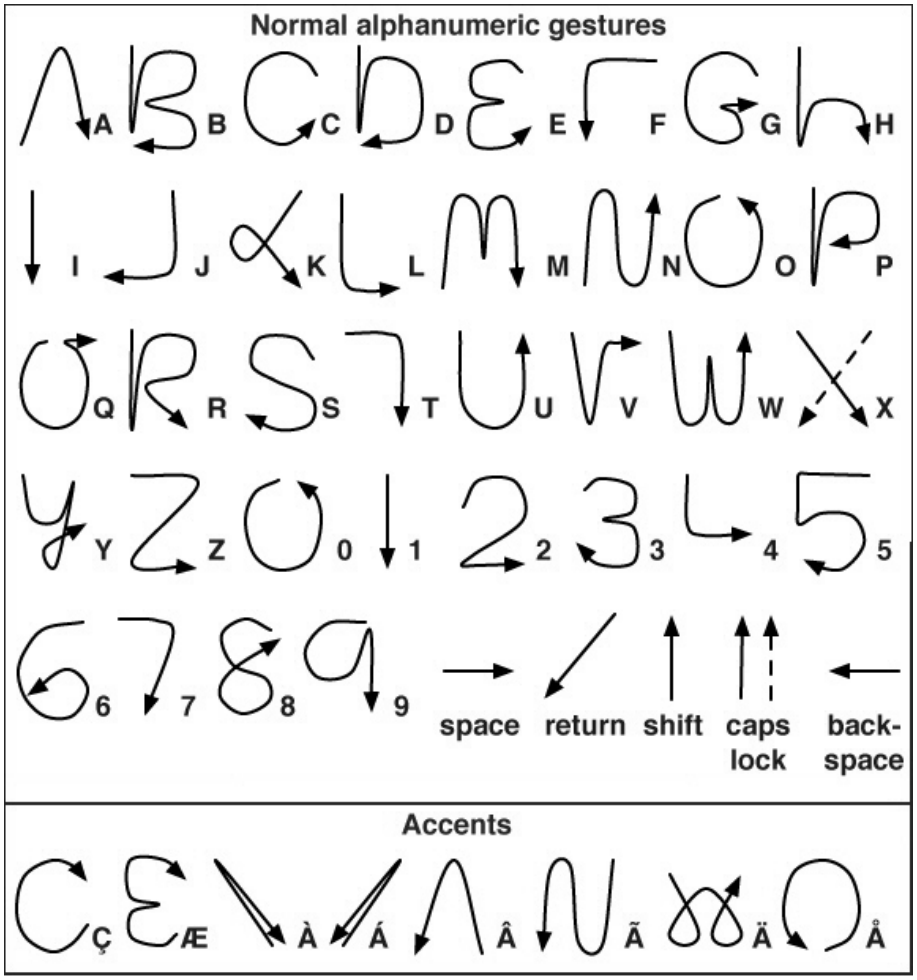
8.2.6 Beidhändige Gesten

Die Verwendung einer zweiten WiiMote oder des Nunchunk-Controllers, der drei zusätzliche Beschleunigungssensoren für die Verwendung in der zweiten Hand

enthält, ermöglicht das Erfassen beidhändiger Gesten. Auf Klassifikationsebene sollte hierfür keine Anpassung nötig sein, da hier die Anzahl der an Gesten beschreibenden Signale beliebig ist. Auf Anwendungsebene muss hierfür ein Modul implementiert werden, das die zusätzlichen Sensorinformationen miterfasst.

Anhang A

PDA-Gesten für Handschriften



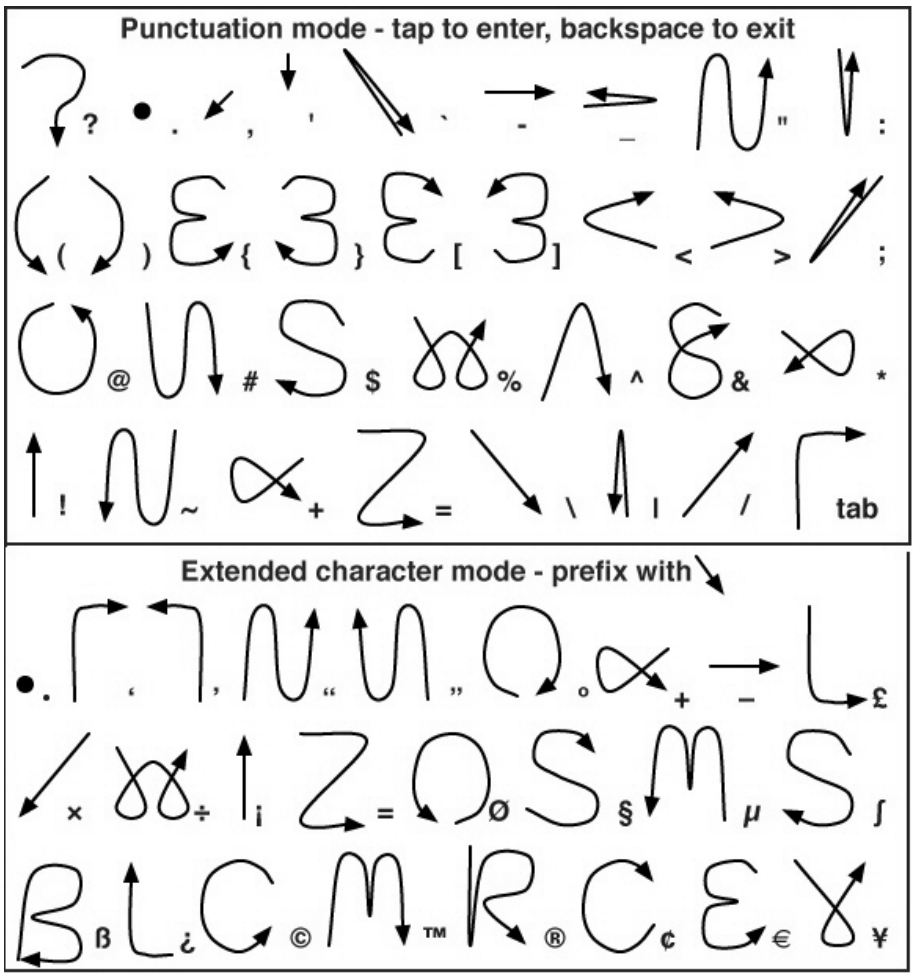


Abbildung A.1: Gesten für die Erkennung von Handschriften auf dem Palm

Literaturverzeichnis

[Ailive, 2006] Ailive (2006). Livemove White Paper. Technical report, AILive Inc. `http://www.ailive.net/papers/LiveMoveWhitePaper_en.pdf`.

[Alexandersson et al., 2006] Alexandersson, J., Richter, K., and Becker, S. (2006). i2home: Benutzerzentrierte Entwicklung einer offenen standardbasierten Smart Home Plattform. Technical report, DFKI.

[Baier et al., 2007] Baier, V., Mosenlechner, L., and Kranz, M. (2007). Gesture Classification with Hierarchically Structured Recurrent Self-Organizing Maps. In *Networked Sensing Systems, 2007. INSS '07. Fourth International Conference on*, pages 81–84.

[Bailador and Guadarrama, 2007] Bailador, G. and Guadarrama, S. (2007). Robust Gesture Recognition using a Prediction-Error-Classification Approach. In *Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International*, pages 1–7. IEEE Computer Society.

[Benbasat and Paradiso, 2001] Benbasat, A. Y. and Paradiso, J. A. (2001). An Inertial Measurement Framework for Gesture Recognition and Applications. In *Gesture Workshop*, pages 9–20.

[Berton et al., 2006] Berton, A., Kaltenmeier, A., Haiber, U., and Schreiner, O. (2006). Speech Recognition. In Wahlster, W., editor, *SmartKom: Foundations of Multimodal Dialogue Systems*, pages 85–107. Springer-Verlag.

[Besser et al., 2007] Besser, J., Neßelrath, R., Pfalzgraf, A., Schehl, J., Steigner, J., and Pfleger, N. (2007). Selection of Technical Components and System Specification. Technical report, DFKI. i2home Deliverable D6.1.

[Billinghurst, 2002] Billinghurst, M. (2002). *Human Input to Computer Systems: Theories, Techniques and Technology*, chapter Gesture Based Interaction. William Buxton.

[Bishop, 1995] Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*, chapter The Multi-layer Perceptron, pages 116–137. Oxford University Press.

[Boehm et al., 1994] Boehm, K., Broll, W., and Sokolewicz, M. (1994). Dynamic Gesture Recognition using Neural Networks. A Fundament for Advanced Interaction Construction. *Proceedings of the SPIE—The International Society for Optical Engineering*, pages 336–46.

[Bruegge et al., 2007] Bruegge, B., Teschner, C., Lachenmaier, P., Fenzl, E., Schmidt, D., and Bierbaum, S. (2007). Pinocchio: Conducting a Virtual Symphony Orchestra. In Inakage, M., Lee, N., Tscheligi, M., Bernhaupt, R., and Natkin, S., editors, *Advances in Computer Entertainment Technology*, pages 294–295. ACM.

[Chalmann and Thiel, 2002] Chalmann, E. and Thiel, S. (2002). Sprache und Gestik. student seminar, `http://www.techfak.uni-bielefeld.de/ags/wbski/lehre/digiSA/KommIntelligenz/chalman_thiel.pdf`.

[Encyclopedia, 2003] Encyclopedia (2003). *Encyclopedia Americana*. Grolier Academic Reference.

[Engel and Pfleger, 2006] Engel, R. and Pfleger, N. (2006). *SmartKom: Foundations of Multimodal Dialogue Systems*, chapter Modality Fusion, Integrating the Modality Hypotheses, pages 223–236. Springer-Verlag.

[Holmes et al., 1994] Holmes, G., Donkin, A., and Witten, I. H. (1994). Weka: A Machine Learning Workbench. Technical report, Department of Computer Science,University of Waikato.

[Kipp, 2004] Kipp, M. (2004). *Gesture Generation by Imitation - From Human Behavior to Computer Character Animation*. PhD thesis, Universität des Saarlandes.

[Kohavi et al., 2006] Kohavi, R., Sommerfield, D., and Dougherty, J. (2006). Data Mining using MLC++, A Machine Learning Library in C++. In *Proceedings of the 8th International Conference on Tools with Artificial Intelligence (ICTAI '96)*, page 234. IEEE Computer Society.

[Kurtenbach and Hulteen, 1990] Kurtenbach, G. and Hulteen, E. (1990). *The Art of Human-Computer Interface Design*, chapter Gestures in Human Computer Communication, pages 309–317. Addison-Wesley.

[Mäntyjärvi et al., 2004] Mäntyjärvi, J., Kela, J., Korpipää, P., and Kallio, S. (2004). Enabling Fast and Effortless Customisation in Accelerometer Based Gesture Interaction. In *MUM '04: Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia*, pages 25–31. ACM.

[Mead, 1964] Mead, G. H. (1964). *Geist, Identität und Gesellschaft aus der Sicht des Sozialbehaviorismus*. Suhrkamp-Verlag.

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine Learning*, chapter Artificial Neural Networks, pages 86–131. McGraw-Hill Companies.

[Mäntylä et al., 2000] Mäntylä, V.-M., Mäntyjärvi, J., and Seppänen, T. (2000). Hand Gesture Recognition of a Mobile Device User. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, pages 281–284. `http://www.mediateam.oulu.fi/publications/pdf/106.pdf`.

[Perng et al., 1999] Perng, J., Fisher, B., Hollar, S., and Pister, K. (1999). Acceleration Sensing Glove. In *Wearable Computers, 1999. Digest of Papers. The Third International Symposium on*, pages 178–180. I.E.E.E.Press.

[Platt, 1998] Platt, J. (1998). Fast Training of Support Vector Machines using Sequential Minimal Optimization. In Schölkopf, B., Burges, C., and Smola, A. J., editors, *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press.

[Racky et al., 2006] Racky, J., Lützeler, M., and Röttger, H. (2006). The Sense of Vision: Gestures and Real Objects. In Wahlster, W., editor, *SmartKom: Foundations of Multimodal Dialogue Systems*, pages 153–165. Springer-Verlag.

[Schölkopf and Smola, 2002] Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels*. The MIT Press.

[Schlömer et al., 2008] Schlömer, T., Poppinga, B., Henze, N., and Boll, S. (2008). Gesture Recognition with a Wii Controller. In *TEI '08: Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 11–14, New York, NY, USA. ACM.

[Starner et al., 1998] Starner, T., Weaver, J., and Pentland, A. (1998). Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1371–1375. IEEE Computer Society.

[Stern et al., 2006] Stern, H., Wachs, J., and Edan, Y. (2006). Optimal Hand Gesture Vocabulary Design Using Psycho-Physiological and Technical Factors. In *Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International Conference on*, pages 257–262.

[Torres, 2007] Torres, A. (2007). Gesture-Based Interaction. student seminar, `ls12-www.cs.uni-dortmund.de/~fink/lectures/SS07/intelligent-spaces/gestures.pdf`.

[Turk, 2002] Turk, M. (2002). *Handbook of Virtual Environments: Design, Implementation, and Applications*, chapter Gesture Recognition, pages 223–238. Lawrence Erlbaum Associates Inc,US.

[Wagner et al., 2006] Wagner, S., Alefs, B., and Picus, C. (2006). Framework for a Portable Gesture Interface. In *Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International Conference on*, pages 275–280.

[Wahlster, 2003] Wahlster, W. (2003). Towards Symmetric Multimodality: Fusion and Fission of Speech, Gesture, and Facial Expression. In Günter, A., Kruse, R., and Neumann, B., editors, *KI 2003: Advances in Artificial Intelligence. Proceedings of the 26th German Conference on Artificial Intelligence*, pages 1–18.

[Westeyn et al., 2003] Westeyn, T., Brashear, H., Atrash, A., and Starner, T. (2003). Georgia Tech Gesture Toolkit: Supporting Experiments in Gesture Recognition. In *Proceedings of the 5th international conference on Multimodal interfaces*, pages 85–92. ACM.

[Wilsona and Wilson, 2004] Wilsona, D. and Wilson, A. (2004). Gesture Recognition Using the Xwand. Technical report, Assistive Intelligent Environments Group Robotics Institute Carnegie Mellon University and Microsoft Research. `http://www.cs.cmu.edu/~dwilson/papers/xwand.pdf`.

[Witten and Frank, 2000] Witten, I. H. and Frank, E. (2000). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufman.